

版权注意事项：

- 1、书籍版权归作者和出版社所有
- 2、本PDF仅限用于个人获取知识，进行私底下的知识交流
- 3、PDF获得者不得在互联网上以任何目的进行传播
- 4、如觉得书籍内容很赞，请购买正版实体书，支持作者
- 5、请于下载PDF后24小时内删除本PDF。



微信轻应用的先驱者，丰富的项目案例，
由浅至深，一步步带您掌握企业号开发。



实战：八个实际项目案例，从基础入门到高级应用，手把手教你成为微信大牛

简单：30天，上千行核心代码让你精通微信企业号开发

丰富：AngularJS、ECharts、Qpid、WebSocket、Servlet等10余种技术与微信的结合

创新：类似微信小程序的单页面应用开发讲解



微信企业号开发 完全自学手册

牟云飞
编著



牟云飞

高级研发工程师，烟台海颐软件股份有限公司产品经理。参与众多项目开发，具有丰富的项目实战经验，同时也是微信企业号开发的先行者，在实际开发过程中，积累了丰富的知识和经验。

微信企业号开发 完全自学手册

牟云飞
编著

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书是微信公众平台企业号开发较全面、系统的一本书，以实战开发为原则，讲解微信各个模块的开发使用，以实例引导企业号的开发与运用，以 Struts、Hibernate、Servlet、HttpClients、JSP、Ajax、jQuery 等热门技术实现微信 Light App 的开发，通过 QPID、代理服务、页面有效期等方式实现数据的安全交互。除此之外，对 SPA 单页面应用框架如何在微信中运用也做了详细介绍。

本书共 11 章，涵盖的主要内容有：微信公众号概述、企业号的发展与注册、配置微信开发环境、JCE 安全策略、微信企业号开发基础知识、主动推送模式、被动回调模式、企业会话模式、JSAPI 模式、通讯录管理、语音导航、腾讯地图使用、WebSocket 微信开发、微信单页面应用、QPID、前置机数据安全访问、企业资讯、微信考勤等。

本书由简入深，实用性较强，即便没有微信开发经验的读者，也能够一步步学习微信开发，学会每个接口的调用及问题处理。有公众号开发经验的读者，则可以重点阅读 JSAPI 和数据安全章节，丰富企业号应用，解决微信 SPA 物理回退、语音导航等问题。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目 (CIP) 数据

微信企业号开发完全自学手册 / 牟云飞编著. —北京：电子工业出版社，2017.3
ISBN 978-7-121-30809-3

I. ①微… II. ①牟… III. ①移动终端—应用程序—程序设计—手册 IV. ①TN929.53-62

中国版本图书馆 CIP 数据核字 (2017) 第 011055 号

策划编辑：张月萍

责任编辑：安 娜

印 刷：北京京科印刷有限公司

装 订：三河市良远印务有限公司

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱

邮编：100036

开 本：787×1092 1/16 印张：29

字数：736 千字

版 次：2017 年 3 月第 1 版

印 次：2017 年 3 月第 1 次印刷

定 价：76.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：(010) 88254888，88258888

质量投诉请发邮件至 zltz@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

推荐序

随着移动互联网的迅猛推广，尤其是微信等移动社交平台的快速普及，企业运营协作模式也在发生深刻变化，企业信息化走向移动已经成为十分紧迫的课题。诞生于桌面 PC 时代的企业信息化目前还主要应用于桌面环境，移动化可以将信息接入从桌面向智能终端大大延伸，从而突破固有的终端种类、接入时间和地点的局限性，可以实现企业信息化真正的无缝闭环，这无疑是企业信息化发展历程中的一次质的飞跃。

企业移动信息化的实现途径多种多样，从最初的 WAP 网站方式到后来的智能 App 模式，再到轻应用模式，技术实现方式背后蕴藏着各种企业信息化要素的平衡和优化，这些要素包括用户体验、开发成本、企业信息安全、推广效率，等等。微信企业号正是可以满足这些要素的优秀解决方案，因此甫一推出就立刻受到了广泛的关注和认可。

在用户体验方面，由于企业信息化应用绝大多数涉及的只是信息的浏览和表单的处理，对用户体验的方面要求并不高，因此微信平台提供的轻应用完全可以胜任。从开发成本上考虑，由于微信企业号的开发采取的是跨平台的网页开发技术，而开发跨平台应用，相较于原生 App 开发无疑可以节省大量的开发和测试成本，对于项目来说，也就意味着可以在较短的时间内实现应用上线，从而迅速产生实际效益。

仅有项目开发的“多快好省”还不行，在数据成为企业新的重要资产的今天，互联网环境下的信息安全成为企业在部署移动化之前不得不考虑的前提。传统的移动信息安全一般要借助于移动设备管理 MDM 等系统级平台，项目投资大，对于移动设备的侵入性也非常大，对于中小型企业来说，往往难以承受。针对信息安全，微信企业号提供了相对轻量级的安全架构，将内部相对封闭的内部组织架构与个人微信号实现绑定，通过企业号后台可以对进入企业号的微信号进行认证、绑定以及后续的注销，具备基本的用户管理能力，而进一步更严格的认证措施则可以通过企业号的二次开发来实现。

最后再来看一下微信企业号的推广效率。由于微信本身已经成为覆盖绝大多数企业员工的社交平台，企业往往也已经建立了各种微信群或微信公众号，借助于这些传播渠道，微信企业号可以很轻易地获得推广，而且绑定动作相对于 App 的安装来说轻量且优雅，也不存在 App 后续的升级更新问题。

综合以上特征，个人认为微信企业号对于信息敏感性不太高的中小企业来说无疑是最适合的企业移动信息化扩展平台。通过在微信企业号上进行一定的二次开发，就可以轻易地使企业内部的信息化系统具备移动化能力。

我们海颐软件正是这样一家中等规模的软件企业，并较早成为了微信企业号的用户。本书的作者正是我们微信企业号的主要开发者。在实际开发过程中，他积累了丰富的知识和经验。相信借助于本书，您可以绕开大部分的困扰和陷阱，帮助您直达目标，迅速构建出令人满意的微信企业号应用来。

李锐

烟台海颐软件股份有限公司 副总经理

为什么要写这本书

智能手机的日渐普及不断地推动着移动互联网在各行业的应用,众多的 App 琳琅满目,App 开发也从最初的 Native App 开发,发展到 Native App、Web App 等多种开发技术。开发越来越容易,各类客户需求的分散,导致 App 越来越多,大量功能单一的应用被搁置,成为“僵尸应用”。越来越多的用户将视线聚集到微信、QQ、新浪微博等超级应用中,在超级应用倍受关注的形势下,Light App 应运而生。Light App 又被称为轻应用、微应用,是一种无须下载、即搜即用的全功能 App,既有媲美甚至超越 Native App 的用户体验,提升用户群体,又具备 Web App 快速开发节约开发成本等特性,前景更加广阔。

微信公众号是腾讯公司在微信的基础上推出的,属于 Light App 的范畴,使广大微信用户无须下载便能够借助微信直接享受个人或企业提供的各类服务。对于企业推广和发展来说,企业可以通过订阅号、服务号打造一个基于微信的服务或推广平台;而对企业内部,企业能够通过微信新推出的企业号实现对内部管理系统的集成,包括:人力资源管理系统、报销管理系统、企业论坛、新闻通知公告、即时通信系统、OA 协同办公等系统,使各类系统移动化,提高工作效率。

企业号能够高效地帮助政府、企业及组织构建自己独有的生态系统,随时随地连接员工、上下游合作伙伴及内部系统和应用,实现业务及管理的互联网移动化。2014 年 9 月企业号进行公测,2015 年笔者因工作需要开始进行微信企业号开发,当时市面上基本没有企业号开发的相关文章,笔者先后完成多个企业号项目开发,编写数个微信企业号建设方案,并在 CSDN 发布了几篇博文,收到许多读者和企业的来信。随着企业号近两年的发展,越来越多的企业想开发企业号,企业号开发人员也受到软件公司的青睐。

为了把微信公众平台开发经验以及企业号的运用更详细、系统地分享出来,笔者在源智图书李幸编辑的鼓励下编写了这本书,希望认识更多的 IT 人才。

本书内容及知识体系

第 1 篇 微信企业号概述及开发基础知识(第 1~2 章)

本篇介绍了微信公众平台企业号概述以及微信企业号开发环境的配置和开发基础知识,

主要包括微信公众号的区别、企业号的发展与注册、配置微信开发环境、JCE 安全策略的调整、微信调试工具的安装与使用、HttpClient 服务请求调用、域名发布使用以及 Properties 文件配置等。

第 2 篇 典型模块开发（第 3~7 章）

本篇介绍了微信开发各种模式下接口调用及开发实现，主要包括 AccessToken 申请、Token 缓存处理、各类消息的主动推送、素材管理、开启回调模式、消息的接收与响应、ECharts 运用、语音导航实现、WebSocket 连接实现、SPA 开发、企业 IM 与微信的对接、通讯录异步任务维护以及现场业务上报实现等。

第 3 篇 微信数据安全及数据库基础（第 8~9 章）

本篇主要介绍了微信公众号数据安全访问的方式策略，主要从软件开发角度实现数据的传输，通过识别浏览、OAuth 2.0 身份验证、页面访问有效期、QPID 以及前置机代理服务等方式实现数据的传输。

第 4 篇 项目案例实战（第 10~11 章）

本篇主要以案例的方式介绍了微信企业号应用的开发过程，从应用创建到应用开发实现，一步步带领读者学习企业号开发，学习企业资讯、微信考勤等应用的实现。

适合阅读本书的读者

- 需要全面学习微信企业号开发技术的人员；
- 微信公众号开发技术人员；
- 微信单页面应用开发人员
- Java 程序员；
- Java EE 开发工程师；
- 希望提高微信项目开发水平的人员；
- 专业培训机构的学员；
- 微信企业号应用开发项目经理；
- 需要一本微信功能查询与实现手册的人员。

阅读本书的建议

- 没有微信开发经验的读者，建议从第 1 章顺次阅读并演练每一个实例。
- 有一定微信开发基础的读者，可以根据实际情况有选择地阅读各个模块和项目案例。
- 对于有微信公众号项目经验或者对单页面应用开发有兴趣的读者，可以重点阅读第 5 章 JS-SDK 的相关开发。
- 阅读时建议首先阅读一遍书中的模块和项目案例，然后从 Hello World 写起，“千里之行，始于足下”。大到每个案例，小到每行代码，哪怕简单的变量定义也在 SDK 中书写一遍，这样不仅能够提高代码速度、效率，而且理解起来也会更加深刻、容易。

致谢

感谢微信创始人张小龙先生及其团队创造了微信这一优秀的平台；
感谢海颐软件李锐、宋庆伟、于洋提供的微信公众账号开发机遇；
感谢徐国智、李明、马金刚在技术上的启蒙与指导；
感谢于春洋在我写书期间在生活上给予的鼓励与帮助；
感谢身边的同事以及广大 IT 网友对这本书的支持与鼓励。

| 目 录 |

第一篇 从零开始学企业号

第 1 章 微信公众平台——认识企业号	2
1.1 微信企业号简介	2
1.1.1 平台发展历程	2
1.1.2 企业号定位	3
1.1.3 与订阅号、服务号区别	3
1.1.4 企业号应用	4
1.2 企业号注册	5
1.2.1 基本信息	5
1.2.2 邮箱激活	5
1.2.3 选择类型	6
1.2.4 信息登记	7
1.2.5 公众号信息	10
1.2.6 绑定管理员	11
1.2.7 增加管理员	11
1.2.8 认证	13
1.3 应用创建	14
1.3.1 进入应用中心	14
1.3.2 选择应用类型	15
1.3.3 填写应用信息	15
1.3.4 完成应用创建	16
第 2 章 平台开发基础入门	17
2.1 JDK 及 JCE 补丁部署	17
2.1.1 安装 JDK	17
2.1.2 环境变量	19
2.1.3 JCE 安全策略补丁	21
2.2 开发环境	22
2.2.1 安装 MyEclipse	22
2.2.2 绑定服务器	24
2.2.3 调整编译环境	26
2.2.4 微信 web 开发工具	27

2.3	HttpClient 使用技巧.....	29
2.4	URLConnection 使用技巧.....	32
2.5	Properties 配置文件使用.....	36
2.6	接口调试工具.....	37
2.7	发布外网服务.....	38
2.8	公众平台消息模式.....	39
2.9	微信企业号入门 Hello World.....	40

第二篇 微信企业号开发核心技术

第 3 章	主动调用模式.....	46
3.1	主动调用模式介绍.....	46
3.2	申请 AccessToken.....	47
3.3	AccessToken 的缓存处理.....	50
3.4	主动调用频率限制.....	53
3.5	信息推送.....	53
3.5.1	接口说明.....	54
3.5.2	推动文本消息.....	56
3.5.3	推送图片消息.....	61
3.5.4	推送语音消息.....	62
3.5.5	推送视频消息.....	66
3.5.6	推送文件消息.....	70
3.5.7	推送新闻消息.....	73
3.5.8	推送永久图文消息.....	79
3.5.9	管理端推送消息.....	86
3.6	素材管理.....	87
3.6.1	接口说明.....	87
3.6.2	上传临时素材文件.....	87
3.6.3	获取临时素材文件.....	90
3.6.4	上传永久素材（非图文素材）.....	92
3.6.5	上传永久素材（图文素材）.....	93
3.6.6	获取永久素材（非图文素材）.....	97
3.6.7	获取永久素材（图文素材）.....	98
3.6.8	删除永久素材.....	99
3.6.9	修改永久图文素材.....	100
3.6.10	获取素材总数.....	101
3.6.11	获取素材列表.....	102
3.6.12	管理端素材维护.....	104
3.7	企业号应用管理.....	105
3.7.1	获取企业号应用.....	105
3.7.2	设置企业号应用.....	107

3.7.3	获取应用概况列表.....	108
3.7.4	管理端应用管理.....	109
3.8	主动模式自定义菜单.....	110
3.9	信息自动回复.....	111
3.10	案例：业务派单.....	113
第 4 章	被动回调模式.....	117
4.1	被动回调模式介绍.....	117
4.2	开启回调模式.....	119
4.3	加密/解密算法.....	123
4.4	被动模式自定义菜单.....	125
4.4.1	限制与说明.....	125
4.4.2	创建菜单.....	127
4.4.3	删除菜单.....	132
4.4.4	获取菜单列表.....	133
4.4.5	管理端菜单维护.....	134
4.5	接收消息 Dom 解析.....	135
4.6	消息响应 Xstream 转换.....	138
4.7	接收普通消息.....	141
4.7.1	接口说明.....	141
4.7.2	接收文本消息.....	145
4.7.3	接收图片消息.....	146
4.7.4	接收音频消息.....	147
4.7.5	接收位置消息.....	148
4.7.6	接收小视频消息.....	149
4.7.7	接收链接消息.....	151
4.7.8	接收视频消息.....	152
4.8	接收事件消息.....	153
4.8.1	接口说明.....	153
4.8.2	接收关注/取消关注事件.....	155
4.8.3	接收地理位置事件.....	157
4.8.4	接收进入应用事件.....	158
4.8.5	接收菜单事件.....	159
4.8.6	接收异步任务完成事件.....	166
4.9	被动响应消息.....	167
4.9.1	接口说明.....	167
4.9.2	被动响应文字消息.....	169
4.9.3	被动响应图片消息.....	171
4.9.4	被动响应音频消息.....	173
4.9.5	被动响应视频消息.....	175
4.9.6	被动响应图文消息.....	177
4.10	案例：企业通讯录快速搜索.....	180

第 5 章 JSAPI 模式	192
5.1 JSAPI 模式介绍	192
5.2 页面接口引入	193
5.2.1 配置“可信域名”	193
5.2.2 引入微信 JS 文件	194
5.2.3 权限验证	194
5.2.4 验证成功事件	199
5.2.5 验证失败事件	199
5.3 Debug 调试及基础接口说明	199
5.3.1 Debug 调试模式开启	199
5.3.2 判断当前客户端版本是否支持指定 JS 接口	200
5.3.3 接口通用函数	201
5.4 微信 JS-SDK 接口说明	201
5.5 权限接口应用	202
5.5.1 隐藏右上角菜单	202
5.5.2 GPS 定位获取位置信息	204
5.5.3 图片处理接口	205
5.5.4 语音及智能接口	206
5.6 ECharts 在微信中的应用	208
5.6.1 ECharts 简介	208
5.6.2 ECharts 快速接入	208
5.6.3 ECharts 微信应用	210
5.7 微信中的地图语音导航	214
5.7.1 微信内置地图导航	214
5.7.2 腾讯地图语音导航	215
5.7.3 百度地图语音导航	217
5.8 微信 SPA 开发	219
5.8.1 基于 AngularJS 的 onsenUI	219
5.8.2 创建 AngularJS 微信服务	220
5.8.3 SPA 下 JSAPI 模式权限初始化	221
5.8.4 SPA 下获取 OAuth 2.0 成员身份信息	222
5.8.5 解决微信物理回退	223
5.9 微信 WebSocket 开发	224
5.9.1 WebSocket 客户端	224
5.9.2 WebSocket 服务端	226
5.10 微信中的支付宝	228
5.11 常见问题	229
5.12 案例：现场业务上报	232
5.12.1 场景回顾	232
5.12.2 示例代码展示	232

第 6 章 企业会话模式	240
6.1 企业会话模式介绍	240
6.2 开启企业会话	242
6.3 推送聊天信息	245
6.3.1 信息推送接口说明	245
6.3.2 聊天消息体结构说明	247
6.3.3 创建多聊会话	250
6.3.4 修改多聊会话	253
6.3.5 退出多聊会话	255
6.3.6 获取多聊会话信息	256
6.3.7 清除未读会话状态	257
6.3.8 会话消息免打扰	258
6.4 接收聊天信息	260
6.4.1 信息接收接口说明	260
6.4.2 普通消息结构体说明	262
6.4.3 事件消息结构体说明	265
6.5 案例：企业 IM 与微信的对接	267
第 7 章 通讯录管理及异步任务	275
7.1 成员验证关注	275
7.2 部门管理	276
7.2.1 新增部门	276
7.2.2 更新部门	277
7.2.3 删除部门	278
7.2.4 获取部门列表	278
7.3 成员管理	279
7.3.1 新增成员	280
7.3.2 成员扩展属性 extattr	281
7.3.3 维护成员信息	282
7.3.4 删除单个成员	283
7.3.5 批量删除成员	284
7.3.6 获取成员信息	284
7.3.7 获取部门成员	286
7.3.8 获取部门成员及详细信息	287
7.4 异步任务管理	289
7.4.1 上传 CVS 文件	290
7.4.2 全量覆盖部门	292
7.4.3 全量覆盖成员	296
7.4.4 jobid 获取异步任务结果	299
7.4.5 callback 接收异步任务通知	302
7.5 标签管理	305
7.5.1 创建标签	305

7.5.2	新增标签成员	307
7.5.3	删除标签成员	310
7.5.4	获取标签成员	313
7.5.5	删除标签	313
7.6	案例：企业通讯录异步维护	314
第 8 章	数据安全访问策略	321
8.1	OAuth 2.0 身份验证	321
8.1.1	获取 code	322
8.1.2	根据 code 获得成员信息	323
8.2	浏览器类型安全访问	325
8.3	全局验证码变量	326
8.4	页面有效期访问	327
8.4.1	JS 定时任务校验	328
8.4.2	事件校验	329
8.5	QPID 消息队列	330
8.5.1	QPID 消息 Hello World	330
8.5.2	QPID 发送 MAP 消息	333
8.5.3	8080 端口问题	336
8.6	代理服务器	337
8.7	企业号服务 IP 白名单	339
8.8	案例：通过 DMZ 服务器获取内网图片	341
第 9 章	数据库及服务器	348
9.1	常用 SQL 语句	348
9.1.1	查询语句	348
9.1.2	新增语句	350
9.1.3	更新语句	350
9.1.4	删除语句	351
9.2	HQL 语句基础语法	351
9.3	HQL 方言处理	354
9.4	Tomcat 服务器	355
9.4.1	在 SDK 中部署	355
9.4.2	8080 端口号冲突	356
9.4.3	内存调整	358
9.4.4	清理数据缓存	358
9.5	JBoss 服务器	359
9.5.1	JBoss 在 SDK 中安装	359
9.5.2	修改 8080 端口	360
9.5.3	JBoss 内存调整	361
9.5.4	发布缓存处理	363
9.6	WebLogic 服务器	363

9.6.1 域的创建.....	363
9.6.2 WebLogic 在 SDK 中安装.....	367
9.6.3 7001 端口号调整.....	368
9.6.4 服务器缓存清理.....	368

第三篇 综合案例

第 10 章 基础应用——企业资讯.....	370
10.1 创建应用.....	371
10.2 获取开发者信息.....	371
10.3 开发实现.....	372
10.3.1 创建数据库 Table.....	372
10.3.2 生成 PO/VO 实体类.....	374
10.3.3 创建工具类 WxUtil.....	379
10.3.4 创建 Web 服务.....	382
10.3.5 Service 处理 Web 请求.....	384
10.4 开启企业资讯应用回调.....	390
10.5 创建最新资讯菜单.....	391
10.6 本章小结.....	391
第 11 章 更进一步：微信考勤.....	392
11.1 场景回顾.....	393
11.2 腾讯地图引入.....	393
11.2.1 腾讯地图 Key 申请.....	394
11.2.2 腾讯地图 Demo.....	395
11.2.3 腾讯地图坐标转换.....	397
11.3 开发实现.....	397
11.3.1 创建微信工具类.....	398
11.3.2 编写回调服务.....	406
11.3.3 考勤信息实体类.....	408
11.3.4 创建业务层服务类.....	409
11.3.5 服务跳转类.....	415
11.3.6 JSP 考勤打卡 Map 页.....	421
11.3.7 考勤查询 JSP 页.....	426
11.3.8 其他考勤页.....	433
11.4 开启微信考勤回调模式.....	435
11.5 绑定可信域名.....	436
11.6 微信考勤应用菜单.....	437
11.7 本章小结.....	437
附录 A 微信表情转换表.....	438
附录 B 返回码说明表.....	441

| 第一篇 |

从零开始学企业号

第1章 微信公众平台——认识企业号

第2章 平台开发基础入门

第 1 章

微信公众平台——认识企业号

微信，又名 WeChat，是腾讯公司于 2011 年 1 月 21 日推出的一款提供免费即时通信服务的智能终端应用程序，截止到 2015 年第一季度时，微信便已经覆盖中国 90% 以上的智能手机，月活跃用户达到 5.49 亿，用户覆盖 200 多个国家、超过 20 种语言，至今已突破 7 亿用户量，是时下最流行的移动社交软件之一。

在微信的不断发展中，微信公众平台于 2012 年 8 月 23 日正式上线，曾命名为“官号平台”和“媒体平台”，最终命名为“公众平台”，以“再小的个体也有自己的品牌”为主题，形成了线上线下微信互动营销的开放应用平台，吸引了众多企业、个体商户、媒体、开发者等加入微信阵营。随着微信公众平台的发展，微信先后推出订阅号、服务号和企业号，应对不同的微信运营主体（政府、组织、企业、个人、媒体等），到目前为止，各品牌的微信公众账号总数已经超过 800 万个，充分发挥了用户的自身价值，提高了用户黏性，形成了一个不一样的生态循环。

庞大的公众账号群体的产生同样孕育了新的 IT 机遇——微信公众平台，本章的重点是帮助读者了解公众平台，学习微信企业号的注册、认证及基础知识，为读者后面的学习提供有力的帮助。

1.1 微信企业号简介

微信企业号，于 2014 年 9 月 18 日正式开启公测，是微信继订阅号和服务号之后，推出的一款企业级应用，主要面向对象为企业、政府及事业单位、社会化组织，是微信为企业提供的移动应用入口，旨在帮助企业建立员工、上下游供应链与企业 IT 系统间的连接，从而快速、低成本实现高质量的移动轻应用，实现生产、管理、协作、运营的移动化。

1.1.1 平台发展历程

微信公众平台，自公测时起，账号类型分为订阅号、服务号，后期推出第三种账号“企业号”，在期间陆续开放了微信支付、微信推广、微信卡券等功能，下面将简单介绍微信公众平台的几个重要历程：

- 2011 年 01 月，腾讯公司推出微信产品；
- 2012 年 08 月，腾讯公司在微信中增加了微信公众平台功能；
- 2012 年 11 月，微信公众平台开放第三方接口，开启轻量应用的大门；
- 2013 年 03 月，微信公众平台推出自定义菜单内测申请；
- 2013 年 07 月，开放微信支付功能；
- 2014 年 04 月，服务号由每月 1 次改为每月（自然月）4 次；
- 2014 年 07 月，微信公众平台公测推广功能；
- 2014 年 09 月，微信“企业号”正式公测；
- 2014 年 12 月，增加成员进入应用的 callback 事件，方便企业账号管理；
- 2015 年 01 月，增加邀请成员关注接口；
- 2015 年 03 月，企业号针对企业开放管理应用接口及登录授权接口；
- 2015 年 06 月，微信企业号增加微信支付；
- 2015 年 09 月，企业号新增微信摇周边接口；
- 2015 年 10 月，企业号新增微信 JS-SDK 接口——企业号会话；
- 2016 年 01 月，企业号信用等级上线。

至今，企业号已形成了单号（企业号）应用模式和双号（企业号+服务号）应用模式。

注意：目前因投诉较多，微信取消邀请关注功能，具体内容将在第 4 章介绍。

1.1.2 企业号定位

微信企业号，是一款企业级应用，定位为企业实现低成本的移动化办公。帮助企业、政府机关、水、电、气、学校等事业单位和非政府组织建立与员工、上下游合作伙伴及内部 IT 系统间的连接，并能有效地简化管理流程、提高信息的沟通和协同效率、提升对一线员工的服务及管理能力，实现业务操作的移动化，提高消息的时效性。

微信企业号于 2014 年 9 月公测，在此之前，企业业务移动化大多采用手机 App 的方式，为此需要开发多个版本的 App，如 Android（安卓）、iOS（苹果手机）、Windows（微软设备）、Mac（苹果电脑）等，虽然可以通过中间代码打包成不同版本的 App，但项目维护成本与开发成本较企业号轻应用来说，还是严重增加了项目投入及开发成本，并且在系统的升级上，App 的维护也存在各自差异，微信企业号企业级轻应用的推出在大幅度减少企业项目投入与开发成本的同时，快速、低成本地实现了企业生产、管理、协作、运营的移动化。

1.1.3 与订阅号、服务号区别

微信公众平台共有三类账号，分别是企业号、服务号和订阅号。正如 1.1.2 节所介绍的，企业号定位方向为企业级应用，主要面向政府、企业、事业单位和非政府组织，提供内部应用在移动端的载体和出口。企业号首先可以保证关注的人员都是特定管理的对象，非通讯录人员扫码无法加入，需要进行身份验证；其次，企业号可以实现消息的多次发送、业务扩展、交流互动等，同时可以将其他业务系统移动化并收纳到企业号中，实现生产管理、协作运营的移动化，为政府、企业、事业单位和非政府组织内部管理提供更多便利。服务号主要用于向企业和组织

提供更强大的业务服务与用户管理能力，帮助企业快速实现全新的公众服务平台。订阅号则主要为媒体和个人提供一种新的信息传播方式，构建与读者之间更好的沟通与管理模式。

微信公众平台三类账号适用于不同的场景，具体差异如表 1.1 所示。

表 1.1 微信公众平台账号区别

	企业号	服务号	订阅号
关注者身份	只有企业号通讯录成员可以关注	任何微信用户	任何微信用户
消息次数限制	最高每分钟可群发200次	不得超过4条/月	每天群发一条
消息保密	可转发、分享，支持保密消息，防止成员转发	可转发、分享	可转发、分享
高级接口	支持	支持	支持
消息显示方式	好友会话列表首层	好友会话列表首层	订阅号目录中
面向人群	面向企业、政府、事业和非政府组织，实现管理、运营的移动化	面向企业、政府、组织，用以对用户进行服务	面向媒体和个人
定制应用	支持	不支持	不支持



使用技巧：公众账号人员关注是否特定范围、消息发送的次数可作为公众账号选择的重要因素。企业号可以包括多个应用，每个消息型应用类似一个服务号，具有单独的消息和菜单功能。

1.1.4 企业号应用

企业号作为一个企业级移动轻应用，主要使用场景如下：

- 企业出差人员的移动办公，如企业内部资讯、通知公告、微信考勤、微信报销、企业论坛、企业微刊等；
- 企业数据报表，适宜企业管理层实时查看公司业务报表等应用场景；
- 用于企业与上下游合作伙伴、供应商的订单管理、工作协同；
- 用于以移动办公为主要场景的一线员工，如售后服务、一线销售、安保后勤等人员的工作管理与支撑；
- 适宜政府机关、水、电、气、学校、医院等事业单位，以及社会组织同样可以通过企业号简化管理流程，提供信息的时效性，提升组织协同运作效率，如现场业务上报、社区信息、客户抢修实景照片回传等。

此外，企业号可以将企业内部 IT 系统或硬件物理设备与员工微信相连接，实现企业系统的移动化，实现良好的闭环流程。同时，企业号较传统的 B/S、C/S 软件等使用更加方便。在项目投入上，企业号属于企业轻应用，与以往的移动 App 相比，投入、维护成本更低。从使用者角度来看，在提高工作效率的同时，减少了移动设备的空间损耗，更使人乐于接受。

1.2 企业号注册

在了解了微信公众平台和微信企业号及其应用之后，本节将向读者介绍企业号开发的第一步——企业号注册，即了解企业号注册需要的信息，学会如何注册、认证企业号，以及解决公众号管理员绑定问题。

1.2.1 基本信息

企业号注册第一步是基本信息的注册，包括邮箱、密码设置、验证码，这里需要注意的是每个邮箱只能申请一种账号，基本信息注册如图 1.1 所示。

1 基本信息

2 邮箱激活

3 选择类型

① 每个邮箱仅能申请一种帐号：公众号或企业号

邮箱

2454516042@qq.com

作为登录帐号，请填写未被微信公众平台注册，未被微信开放平台注册，未被个人微信号绑定的邮箱

密码

.....

字母、数字或者英文符号，最短8位，区分大小写

确认密码

.....

请再次输入密码

验证码

xcag

XCAG

换一张

☒ 我同意并遵守《微信公众平台服务协议》

注册

图 1.1 企业号基本信息注册

!

注意：注册邮箱需要使用既未被微信公众平台和企业号开放平台注册，也未被个人微信号绑定的邮箱。

1.2.2 邮箱激活

完成基本信息注册之后，接下来就是邮箱激活，微信会向注册邮箱发送一封邮件，邮箱需要在 48 小时内激活链接，否则企业号注册失败，激活过程如图 1.2 和图 1.3 所示。

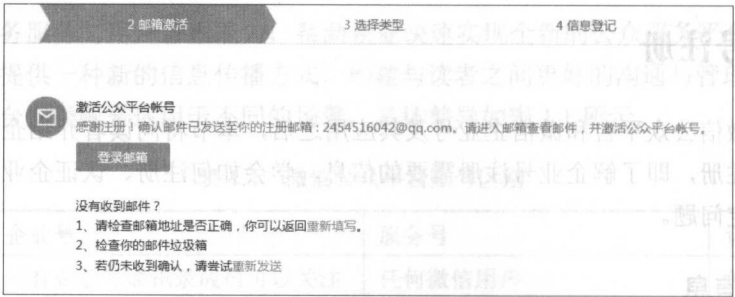


图 1.2 企业号注册邮箱激活

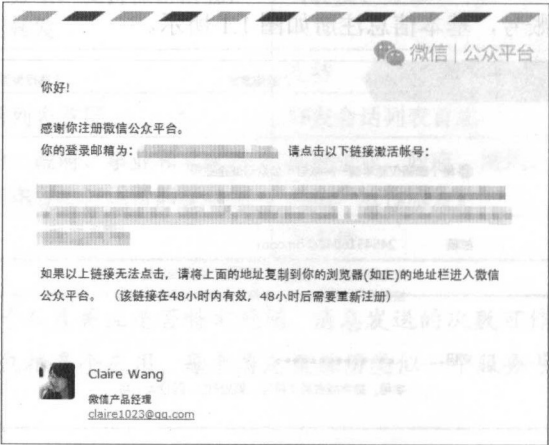


图 1.3 邮箱收到激活邮件

1.2.3 选择类型

邮箱成功激活后，进入选择类型页面，页面分为订阅号、服务号和企业号，如图 1.4 所示。

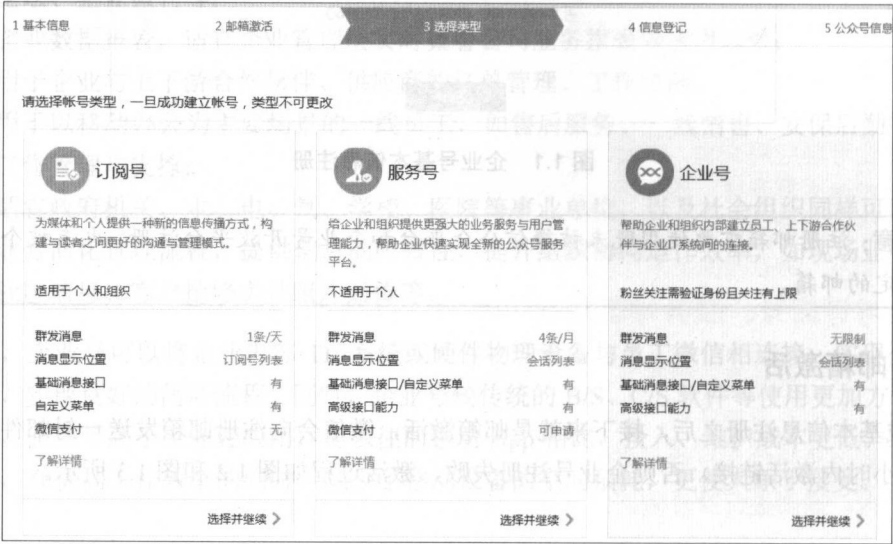


图 1.4 选择公众号类型

注意：账号类型，一旦成功建立将不可修改，请谨慎选择。

这里我们选择企业号并单击“选择并继续”，弹出如图 1.5 所示页面，单击“确定”按钮。

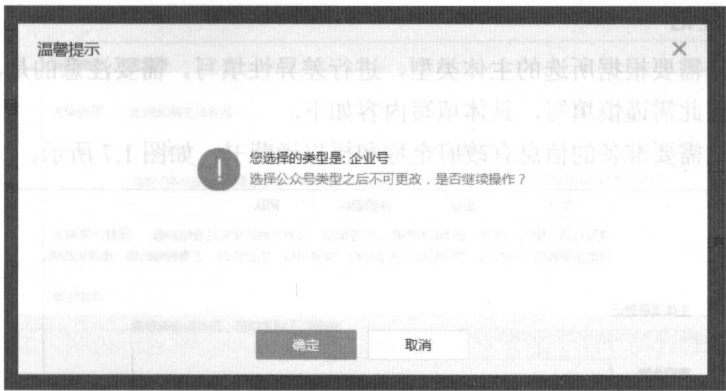


图 1.5 确认选择类型

1.2.4 信息登记

1. 信息登记

信息登记是企业号注册中填写最多的一项，需要根据主体类型分别填写不同的资质信息，在信息登记区域上，主要分为三大区域：选择主体类型、主体信息登记和运营者信息登记。

2. 选择主体类型

在主体类型上，企业号分为政府、企业、注册组织以及团队，如图 1.6 所示。

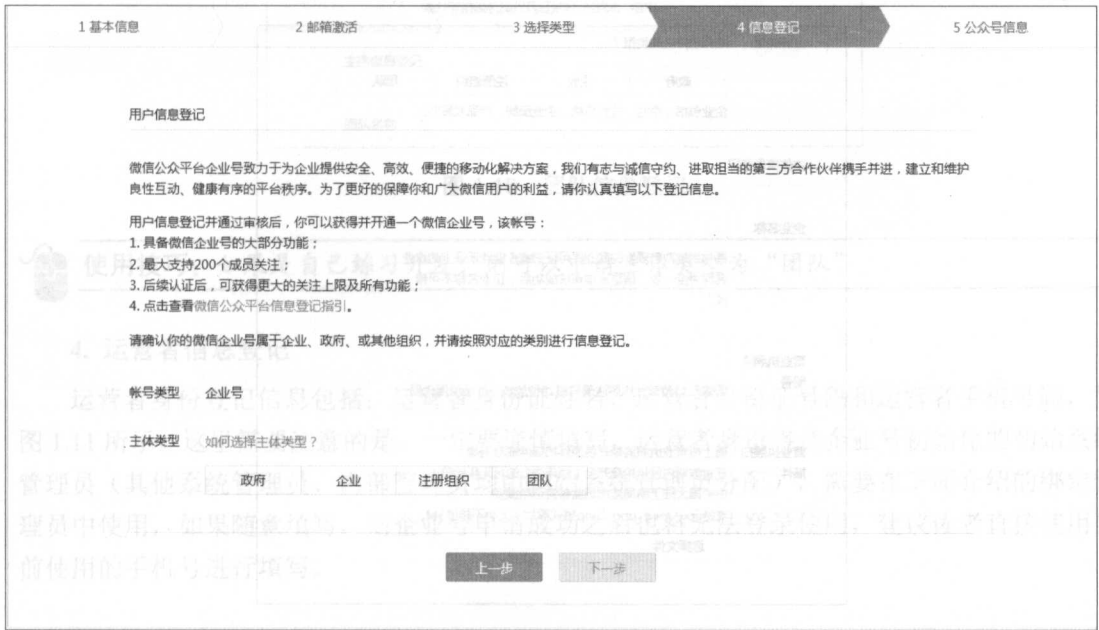


图 1.6 主体类型选择

用户开通后将具备企业号大部分功能，最大支持 200 个成员关注。如果需要扩增人数，则需要企业进行认证，根据提交的资质，微信将分配不同的成员关注最大上限，同时认证后的企业号具备所有功能。

3. 主体信息登记

主体信息登记需要根据所选的主体类型，进行差异性填写。需要注意的是，主体信息提交之后不可修改，因此需谨慎填写，具体填写内容如下。

政府信息登记需要准备的信息有政府全称和授权运营书，如图 1.7 所示。

政府企业注册组织团队

政府包括：国内、各级、各类政府机构、事业单位、具有行政职能的社会组织等。
目前主要覆盖公安机关、党团机构、司法机构、交通机构、旅游机构、工商税务机构、市政机构等。

主体信息登记


政府全称

信息审核成功后，政府全称不可修改

授权运营书 请下载授权运营书按要求填写表格后，上传加盖公章的扫描件
支持.jpg .jpeg .png .bmp格式照片，大小不超过5M。

选择文件

图 1.7 政府信息登记

 说明：政府事业单位的认证审核是由专门的人员进行的。

企业信息登记需要准备的信息有企业名称、营业执照注册号和营业执照扫描件，如图 1.8 所示。

主体类型 如何选择主体类型？

政府企业注册组织团队

企业包括：企业、分支机构、企业品牌、产品或服务。

主体信息登记

企业名称

需与当地政府颁发的商业许可证书或企业注册证上的企业名称完全一致，信息审核审核成功后，企业名称不可修改。

营业执照注册号

请输入15位营业执照注册号或18位的统一社会信用代码

营业执照扫描件 请上传营业执照清晰彩色原件扫描件或数码照
在有效期内且年检章齐全（当年成立的可无年检章）
由中国大陆工商部门或市场监督管理局颁发
支持.jpg .jpeg .png .bmp格式照片，大小不超过5M。

选择文件

图 1.8 企业信息登记



备注：扫描件支持 jpg、jpeg、png 和 bmp 格式的照片，大小不能超过 5MB。

注册组织信息登记需要准备的信息有组织名称、组织机构代码及组织机构代码扫描件，如图 1.9 所示。

帐号类型

企业号

主体类型

如何选择主体类型？

政府

企业

注册组织

团队

其他组织包括：不属于企业、政府的机构等类型的公众帐号。要求机构已办理组织机构代码证。

主体信息登记

组织名称

信息审核成功后，组织名称不可修改

组织机构代码

请输入 9 位组织机构代码，如 12345678-9；或 18 位的统一社会信用代码

组织机构代码扫描件

请上传加盖公章的扫描件

支持 jpg、jpeg、png、bmp 格式照片，大小不超过 5M。

选择文件

图 1.9 注册组织信息登记

团队信息登记需要准备的信息则只有团队姓名，如图 1.10 所示。

主体类型

如何选择主体类型？

政府

企业

注册组织

团队

未办理组织机构代码证的社会团体、组织。

主体信息登记

团队名称

图 1.10 团队信息登记



使用技巧：如果是自己练习开发，则可以申请主体类型为“团队”。

4. 运营者信息登记

运营者身份登记信息包括：运营者身份证姓名、运营者身份证号码和运营者手机号码，如图 1.11 所示。这里需要注意的是，一定要谨慎填写，运营者身份将是企业号初始化的初始系统管理员（其他系统管理员、内部管理员均由初始系统管理员分配），需要在下面介绍的绑定管理员中使用，如果随意填写，则企业号申请成功之后也将无法登录使用，建议读者直接使用当前使用的手机号进行填写。

运营者信息登记

运营者姓名 请填写该公众帐号运营者的姓名，如果名字包含分隔号“.”，请勿省略。

运营者身份证号 请输入运营者的身份证号码。

运营者手机号码 请输入您的手机号码。 [获取验证码]

短信验证码 请输入手机短信收到的6位验证码。 [无法接收验证码？]

运营者身份验证 请先填写组织名称与运营者身份信息

[上一步] [继续]

图 1.11 登记运营者信息

注意：运营者手机号需要注册微信。

1.2.5 公众号信息

完成信息登记之后，企业号的申请注册就进行到了最后一步，公众号信息填写，主要用于填写企业号名称及功能介绍，如图 1.12 所示。

1 基本信息 2 邮箱激活 3 选择类型 4 信息登记 5 公众号信息

帐号名称 企业号开发 10/20
3~20个字符（1个汉字算2个字符），名称一经设置无法更改。

功能介绍 用于企业号开发 7/120

微信认证 enterprise account
[进入企业号]

[返回] [完成]

图 1.12 填写企业号信息

1.2.6 绑定管理员

公众号申请完成之后，将显示一张二维码图片，切勿直接关闭，需要使用第 1.2.4 节中运营者注册的微信号，扫描二维码，并设置密码绑定微信企业号，成功绑定管理员之后，企业号自动添加默认应用“企业小助手”。

如果操作失误，不慎关闭了二维码页面，那么可以通过公众平台进行绑定，账号及密码为 1.2.1 节填写的邮箱及密码，公众平台地址：<https://mp.weixin.qq.com/>。

注意：未绑定管理员的企业号无法登录，也无法使用。

1.2.7 增加管理员

通过官方网址（<https://qy.weixin.qq.com/>）登录企业号管理平台，如图 1.13 所示，单击【设置】|【功能设置】|【权限管理】。



图 1.13 填写企业号信息

进入权限管理界面，在普通管理组中增加“管理组”，这里命名为“开发组”。同时获得 CorpID 以及 Secret，如图 1.14 所示，CorpID 以及 Secret 将用于获取 AccessToken 实现主动模式下消息的推送，详细内容将在第 3 章讲解。通讯录权限用于配置管理组是否对指定的组织具有查看、管理权限。应用权限用于配置管理组是否对指定应用具有发送消息和管理权限。

注意：后期开发中，对于新建应用部分读者发现无法推送信息，原因可能是未对管理员“开发组”中“应用权限”进行设置。

系统管理员为企业号最高管理权限，不建议太多人拥有，配置管理界面如图 1.15 所示。对于普通管理员，在“管理组”创建完成之后，必须分配相应的权限，每个管理组可以分别对企

业号应用设置不同的管理权限，如图 1.16 所示。



图 1.14 创建“管理组”



图 1.15 设置“系统管理员”

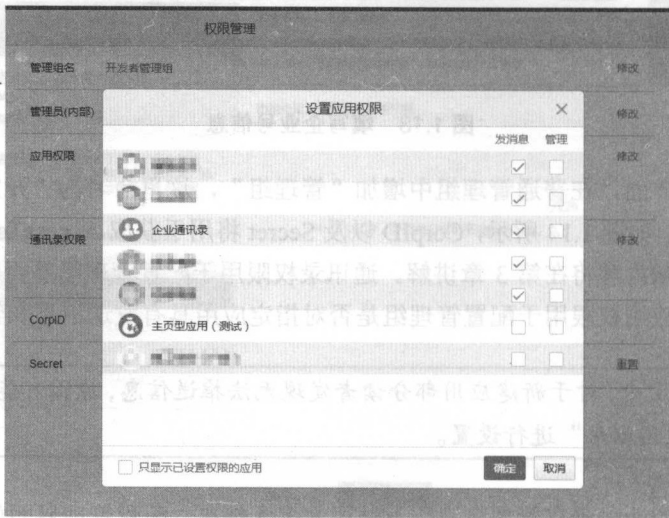


图 1.16 设置管理组权限

1.2.8 认证

企业号的认证工作必须由 1.1 节中企业号的申请人进行认证，认证操作如下。
登录公众号管理平台，单击【设置】|【基本信息】|【进入认证系统】，如图 1.17 所示。

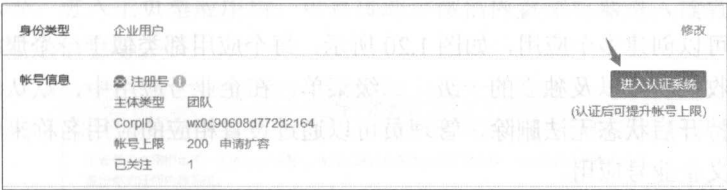


图 1.17 进入认证系统

登录认证系统，开启企业认证，如图 1.18 和图 1.19 所示。

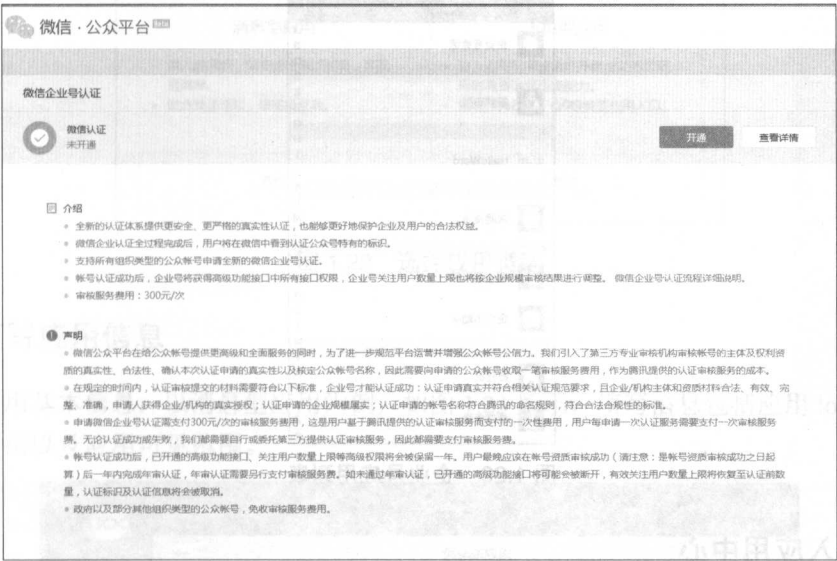


图 1.18 企业号认证界面

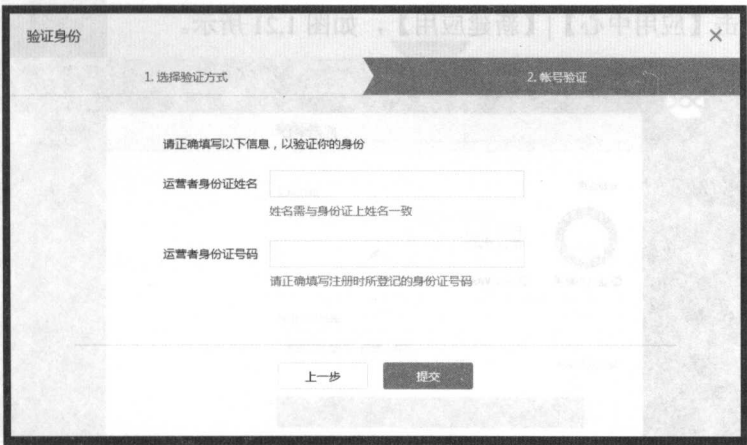


图 1.19 验证身份

备注：企业号申请不花费任何费用，认证则需要 300 元/年的认证费。

1.3 应用创建

每个企业号可以创建多个应用，如图 1.20 所示，每个应用都类似于一个服务号，具有单独的消息发送、接收机制，以及独立的一级、二级菜单。在企业号应用中，默认应用为“企业小助手”，一直保持开启状态无法删除，管理员可以通过设置相应的应用名称来修改应用。本节将演示如何自定义企业号应用。

备注：除企业小助手之外，其他自定义应用管理员能够删除。

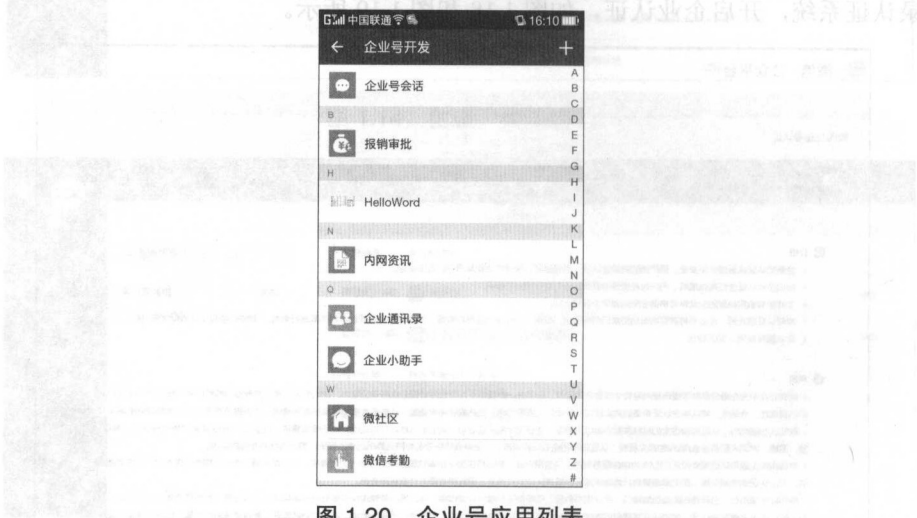


图 1.20 企业号应用列表

1.3.1 进入应用中心

创建企业号应用需要进入应用中心，读者在完成企业号注册之后，使用管理员身份登录企业号管理端，单击【应用中心】|【新建应用】，如图 1.21 所示。



图 1.21 企业号服务端应用中心

1.3.2 选择应用类型

进入创建应用后，第一步需要选择应用类型，如图 1.22 所示。应用类型分为消息型应用和主页型应用两种。进入消息型应用后，将以会话的形式展现（类似服务号），能够自定义菜单，发送、接收信息等。进入主页型应用后，可直接通过微信内置浏览器进入读者设置的页面，无法自定义菜单和发送信息等。

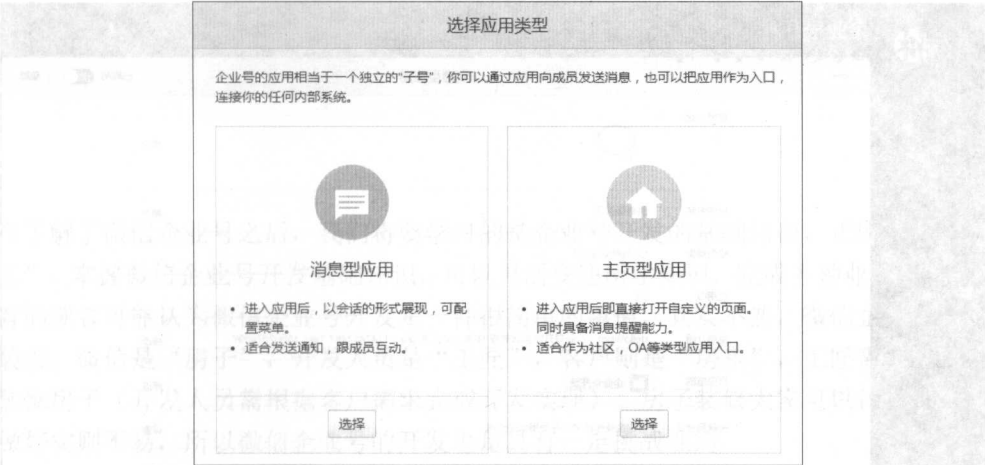



图 1.22 选择应用类型

1.3.3 填写应用信息

填写应用基本信息，以消息型应用为例，如图 1.23 所示，填写信息包括应用 logo、应用名称、功能介绍以及应用可见范围。



图 1.23 填写应用信息

 备注：主页型应用基本信息需要额外填写“主页 URL”。

1.3.4 完成应用创建

完成应用创建后，进入应用配置页面（消息型应用配置页面如图 1.24 所示、主页型应用配置页面如图 1.25 所示），读者可以在配置页面进行其他权限、菜单功能修改。



图 1.24 消息型应用配置页面



图 1.25 主页型应用配置页面

平台开发基础入门

在了解了微信企业号之后，我们将要学习的是企业号开发的基础知识，正所谓“磨刀不误砍柴工”。掌握微信企业号开发基础知识，可以灵活变通所学知识，完成各种业务场景的实现。

有的读者可能认为微信企业号开发是一件很简单的事情，其实不然，微信企业号开发就像房子装修，微信是“房子”，开发人员是“工匠”，客户则是“房东”，工匠需要根据房东的需求装修房子（开发人员需根据客户需求完成开发实现）。房子装修大家可以体会到，看似简单，做好实则不易，所以微信企业号的开发也是具有一定挑战性的。

在技术上，企业号开发可以理解成一个能与微信互动的、免登录的、手机 B/S 网站，如果有兼容手机 B/S 网站开发的读者，可能更容易理解。微信开发不是手机 App 开发，不是 TextView、LinearLayout、Button 等控件操作，而是一个微信消息框架+B/S 手机网站的轻应用，这个网站不需要登录（因为微信已经登录），而且手机网站可以通过微信进行硬件操作，如打开照相机、录音等，还可以利用微信独特的功能，详细内容将在后面介绍。本章将主要介绍微信开发的基础知识，涉及的知识点有：

- JDK 部署及 JCE 安全策略补丁。
- 开发工具及其编译环境。
- HttpClient 使用技巧。
- HttpURLConnection 使用技巧。
- Properties 文件使用。
- 公众平台消息模式介绍。

2.1 JDK 及 JCE 补丁部署

JDK（Java Development Kit），是 Java 语言的软件开发工具包，是整个 Java 开发的核心，包含了 Java 工具和基础的类库。微信企业号开发可以采用多种语言开发，如 Java、C++、PHP 等，这里我们学习的是 Java 开发，本节将向读者介绍 JDK 及 JCE 安全策略补丁的部署。

2.1.1 安装 JDK

本节将演示 JDK 的安装，JDK 可以直接复制原有的 JDK，将其他设备上的 JDK 直接复制

到本机，注意区分 32 位和 64 位系统。也可以通过传统的安装方式，下载 JDK 进行安装，安装步骤如下：

注意：JDK 的版本必须大于等于 1.6，小于 1.6 版本的 JDK 在使用被动回调模式时将出错，被动回调模式将在第 4 章详细介绍。

- 01 下载 JDK 安装包并双击它，如图 2.1 所示。
- 02 显示安装许可协议，如图 2.2 所示，请单击“接受”按钮。

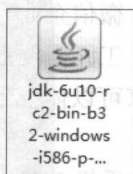


图 2.1 JDK 安装包

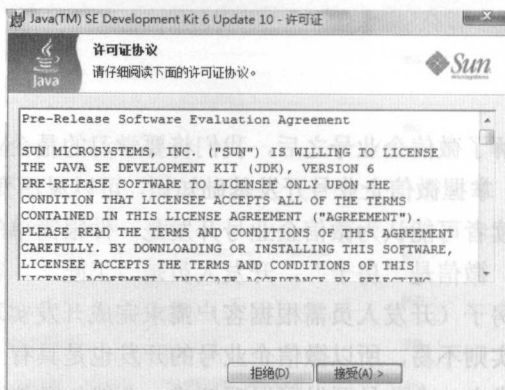


图 2.2 接受 JDK 许可协议

- 03 设置安装路径，也可采用默认路径。32 位系统默认安装路径为 C:\Program Files，64 位系统 64 位的软件默认安装路径为 C:\Program Files，而 64 位系统 32 位的软件则默认安装在 C:\Program Files (x86)目录下，如图 2.3 所示。设置完成后单击“下一步”按钮。
- 04 显示安装进度，在 JDK 安装过程中，将提示安装 JRE（Java Runtime Environment，Java 运行环境，运行 Java 程序所必需的环境的集合，包含 JVM 标准实现及 Java 核心类库），如图 2.4 所示。如果提示安装 JRE，请选择进行安装。

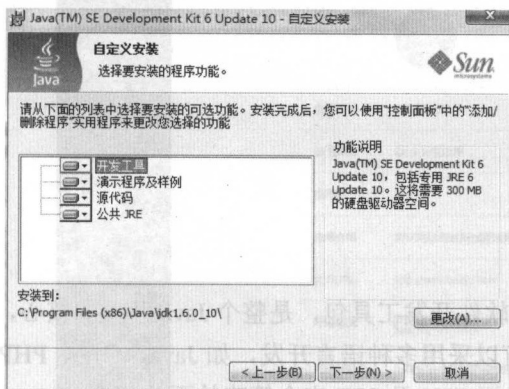


图 2.3 选择安装路径



图 2.4 JDK 安装进度

- 05 安装完成，如图 2.5 所示。

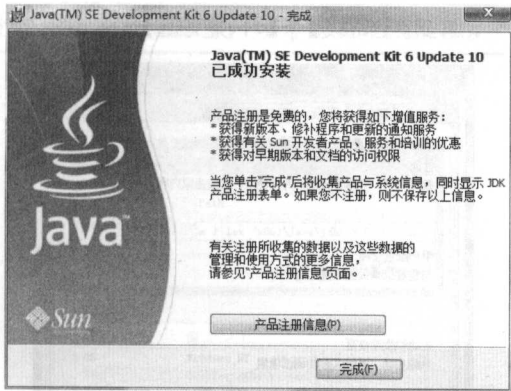


图 2.5 JDK 安装完成

2.1.2 环境变量

环境变量（Environment Variables）一般是指在操作系统中用来指定操作系统运行环境的一些参数，如临时文件夹位置和系统文件夹位置等。在操作系统中，环境变量是一个具有特定名字的对象，它包含了一个或者多个应用程序所将使用到的信息。例如，Java 在 Windows 操作系统中的 Path 环境变量，当要求系统运行一个 Java 程序而没有告诉它 Java 虚拟机所在的完整路径时，系统除了在当前目录下寻找此程序外，还会到 Path 中指定的路径去寻找 Java 虚拟机。用户可通过设置环境变量，来更好地运行进程。

接下来我们开始配置 Java 环境变量，步骤如下。

- 01 在 Win7 系统下，右击【计算机】|【属性】|【高级系统设置】，如图 2.6 所示。在 XP 系统下，右击【我的电脑】|【属性】。



图 2.6 “计算机属性”界面

02 打开“系统属性”，单击【高级】|【环境变量】，如图 2.7 所示。

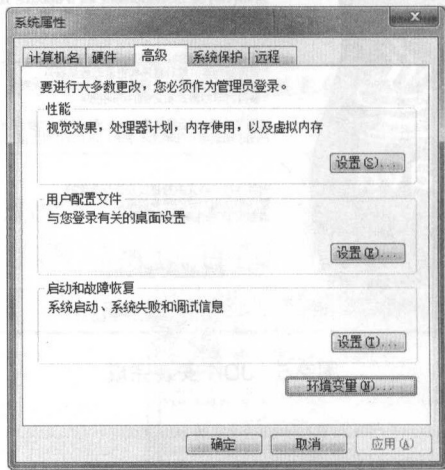


图 2.7 “系统属性”界面

03 环境变量的配置分为三项：JAVA_HOME、CLASS_PATH 和 Path。

JAVA_HOME 为相对路径，后面使用时，可以采用“%JAVA_HOME%\”的方式来使用，如图 2.8 所示。选择【系统变量】|【新建】，新建系统变量，在“变量名”文本框输入“JAVA_HOME”，在“变量值”文本框输入 JDK 的安装路径，如 C:\Program Files (x86)\Java\jdk1.6.0_10，单击“确定”按钮。

CLASS_PATH 为 Java 程序编译成 class 文件之后，需要存放的目录，“.”代表任意目录，与 JAVA_HOME 创建方法一样，创建一个“变量名”为“CLASS_PATH”、“变量值”为“.”的系统变量，如图 2.9 所示。



图 2.8 创建 JAVA_HOME

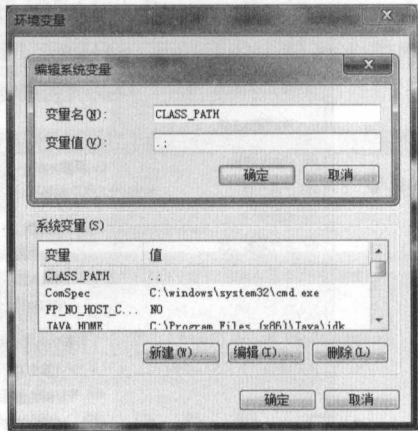


图 2.9 创建 CLASS_PATH

Path 为系统目录，当执行 Java 程序需要虚拟机、工具类时，系统除了在当前目录下寻找外，还将在 Path 中寻找。在“系统变量”中选中 Path，单击“编辑”按钮，添加系统路径，路径之间使用“;”分隔，如图 2.10 所示。

注意：Path 中的变量包括众多系统变量，读者需要谨慎处理。

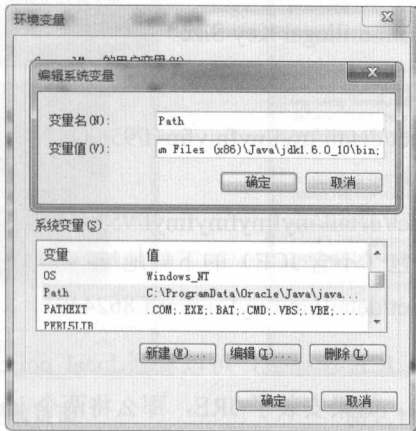


图 2.10 配置 Path

使用技巧：在 Path 中添加 JDK，建议在输入框字符串的最前端填写 JDK 版本信息，避免因 Oracle 等软件影响 JDK 的版本。

04 测试是否修改成功，单击【开始】|【运行】，输入【cmd】，在命令窗口中输入“java -version”并按回车键，将显示出 JDK 版本信息，如图 2.11 所示，表示环境变量修改成功。

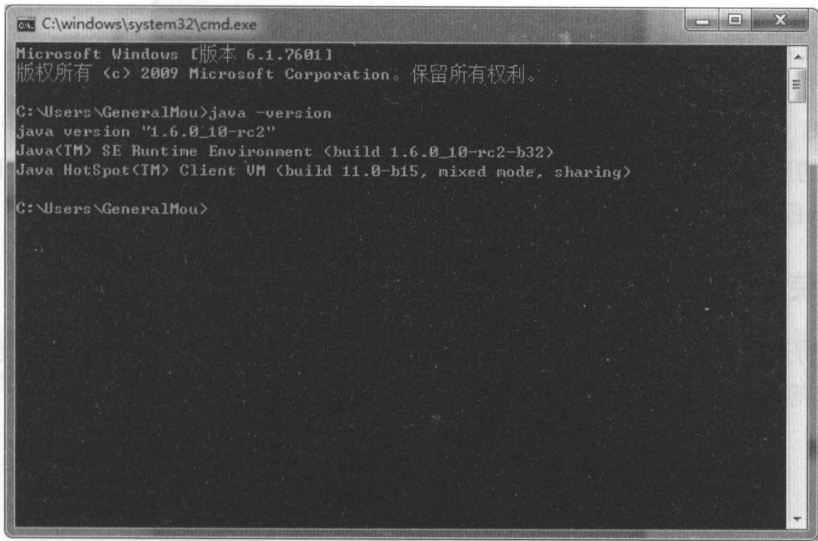


图 2.11 cmd 窗口显示 JDK 版本信息

2.1.3 JCE 安全策略补丁

JCE(Java Cryptography Extension)，用于加密、密钥生成和协商，以及 Message Authentication

Code (MAC) 算法的框架和实现, 提供对对称、不对称、块和流密码的加密支持。支持安全流和密封的对象, 在微信企业号开发过程中, 必须安装 JCE 补丁包, 否则将出现异常 (java.security.InvalidKeyException:illegal Key Size)。

- JDK 6 的 JCE 下载地址:
http://download.csdn.net/detail/myfmyfmyfmyf/9548444
- JDK 7 的 JCE 下载地址:
http://download.csdn.net/detail/myfmyfmyfmyf/9548448
- 微信企业号开发相关包 (不含 JCE) 的下载地址:
http://download.csdn.net/detail/myfmyfmyfmyf/8624491

JCE 补丁安装过程: 解压 JCE 压缩包, 可以看到 local_policy.jar、US_export_policy.jar 和 README.txt, 如图 2.12 所示。如果安装了 JRE, 那么将两个 jar 文件放到%JRE_HOME%\lib\security 目录下覆盖原来的文件; 如果安装了 JDK, 则将两个 jar 文件放到%JDK_HOME%\jre\lib\security 目录下覆盖原来文件即可。





名称	修改日期	类型	大小
 COPYRIGHT.html	2006/11/16 18:10	360 se HTML Do...	3 KB
 local_policy.jar	2006/11/16 18:10	Executable Jar File	3 KB
 README.txt	2006/11/16 18:10	文本文档	9 KB
 US_export_policy.jar	2006/11/16 18:10	Executable Jar File	3 KB

图 2.12 JCE 安全策略补丁包

2.2 开发环境

Java IDE (Integrated Development Environment, Java 集成开发环境), 是用于提供 Java 程序开发环境的应用程序, 由编辑器、调试器、图形化界面等组成的一体化的程序开发软件, 本节将介绍 MyEclipse 的部署及其相关知识。

2.2.1 安装 MyEclipse

企业号的开发不仅可以使⽤ MyEclipse, 如图 2.13 所示, 还可以使⽤企业自定义开发工具以及 Eclipse 等, 接下来我们将学习 MyEclipse 的安装。



图 2.13 MyEclipse 安装文件

- 01 双击 MyEclipse 安装文件, 打开安装首页面, 如图 2.14 所示。
- 02 单击 “Next” 按钮, 进入协议许可页面, 如图 2.15 所示。

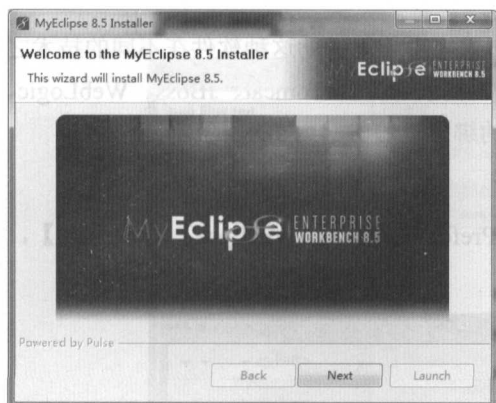


图 2.14 MyEclipse 安装首页面

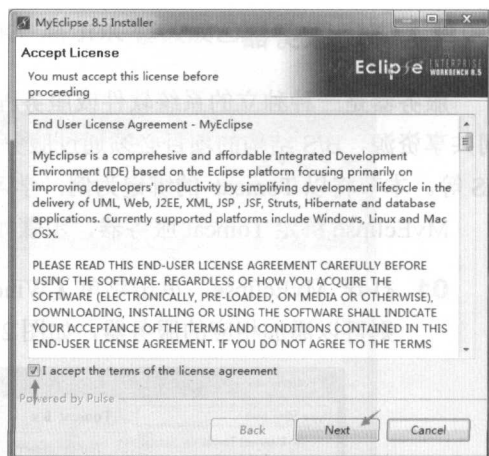


图 2.15 MyEclipse 安装许可

- 03** 接受安装许可，单击“Next”按钮，进入安装目录选择页面，如图 2.16 所示，建议修改安装目录，软件默认安装位置为系统盘。如果系统盘中有过多的软件，将影响电脑性能。

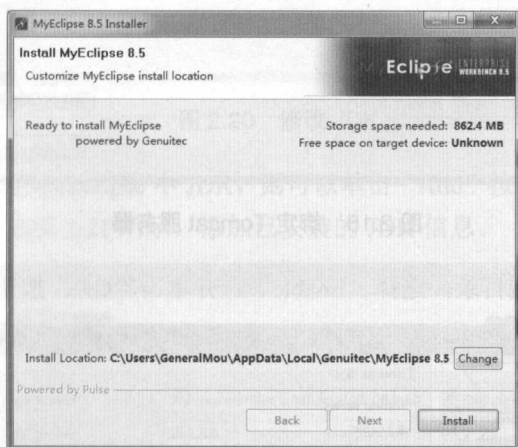


图 2.16 MyEclipse 安装目录选择页面

- 04** 安装过程中，防火墙可能会提示是否允许程序运行，单击“允许”按钮。MyEclipse 安装成功后，将进入 MyEclipse，如图 2.17 所示。

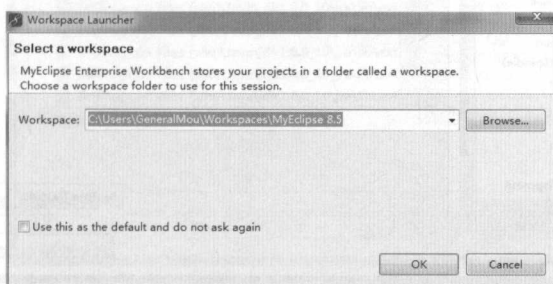


图 2.17 MyEclipse 安装完成

2.2.2 绑定服务器

服务器是一种独立的系统软件或服务程序，分布式应用软件借助这种软件在不同的技术之间共享资源，B/S 结构的项目必须通过服务器进行项目的发布，如 Tomcat、JBoss、WebLogic、IIS 等。本节将以 Tomcat 的绑定为例，学习服务器的绑定及服务器内 JDK 的绑定等。

MyEclipse 绑定 Tomcat 服务器，步骤如下。

- 01 打开 MyEclipse，单击菜单【Window】|【Preferences】|【MyEclipse】|【Servers】，则可以添加相应的服务器，如图 2.18 所示。

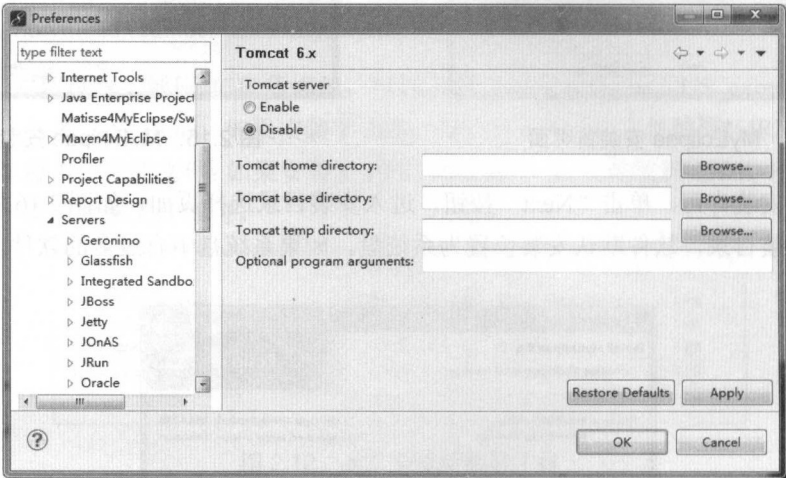


图 2.18 绑定 Tomcat 服务器

- 02 选择 Tomcat 安装目录，选择“Enable”，并单击“OK”按钮，如图 2.19 所示。

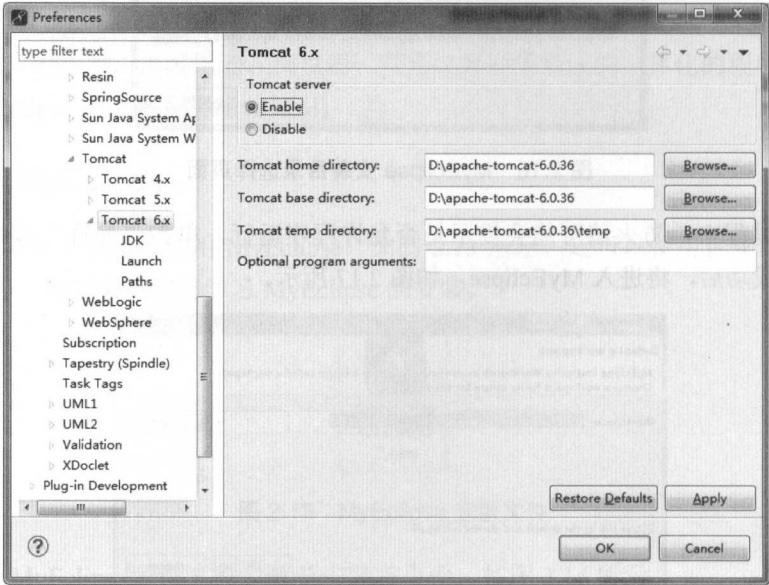


图 2.19 绑定 Tomcat 安装目录

03 绑定成功之后，需要修改 Tomcat 中的 JDK，将 JDK 修改成已经进行过 JCE 安全策略补丁修改的 JDK，如图 2.20 所示。

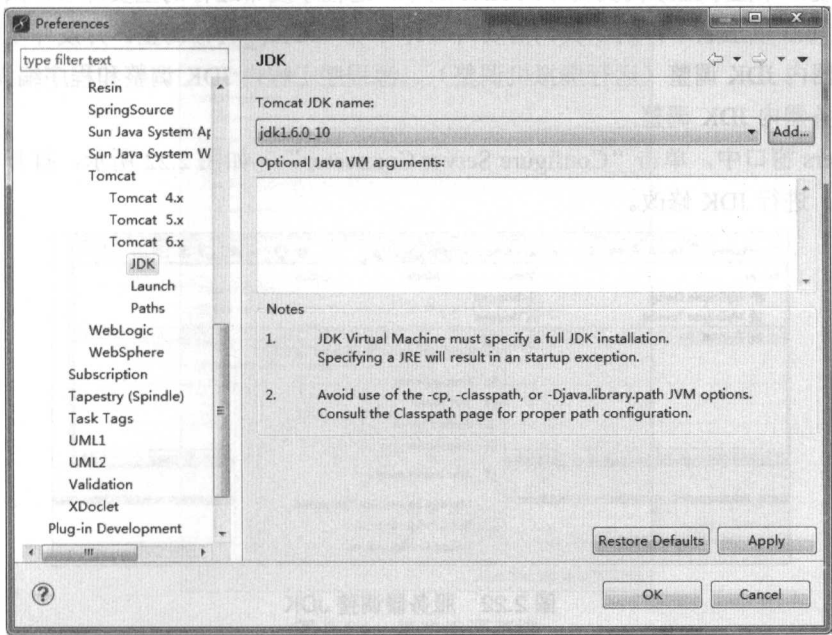


图 2.20 修改 JDK

04 如果尚未配置过 MyEclipse 中 JDK，则可以单击“Add”按钮，打开 MyEclipse 中的添加 JDK 页面，如图 2.21 所示，添加已安装的 JDK 信息。

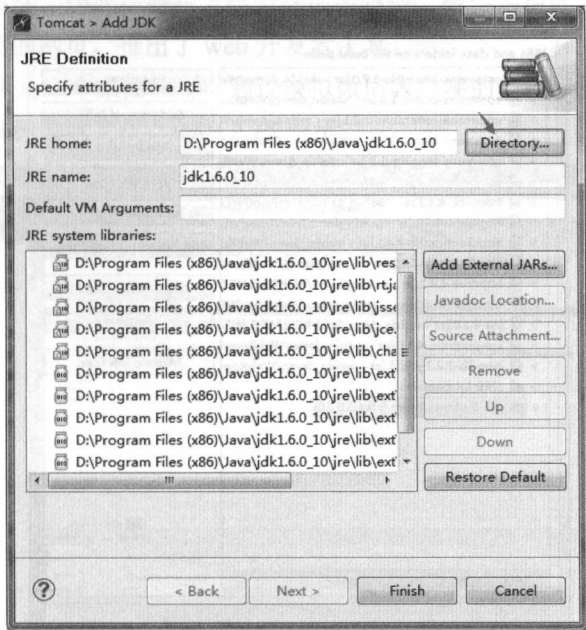


图 2.21 在 MyEclipse 中添加 JDK

2.2.3 调整编译环境

编译环境，即程序虚拟机及编译的 JDK 版本，是程序发布运行的重要环节，微信企业号开发需要 JDK 6.0 以上版本，所以我们需要对编译环境（JDK）进行调整。开发环境的调整主要分为：服务器内 JDK 调整（运行虚拟机调整）、源程序工程内 JDK 调整和程序编译环境调整。

01 服务器内 JDK 调整

在 Servers 窗口中，单击“Configure Server Connector”，如图 2.22 所示，打开相应的服务器配置页面，进行 JDK 修改。

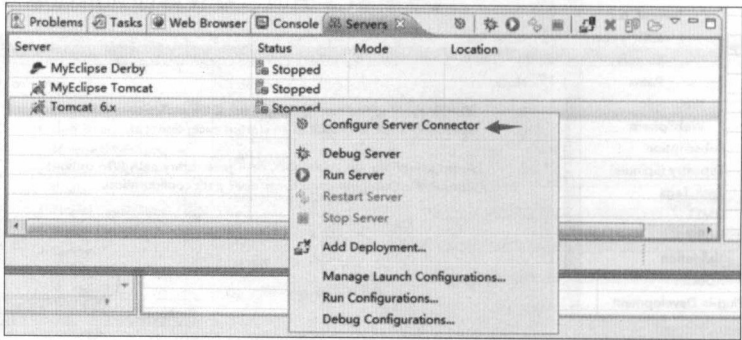


图 2.22 服务器调整 JDK

02 源程序工程内 JDK 调整

在创建的项目工程上，单击“Properties”，然后单击“Java Build Path”，显示如图 2.23 所示页面，修改源工程 JDK。

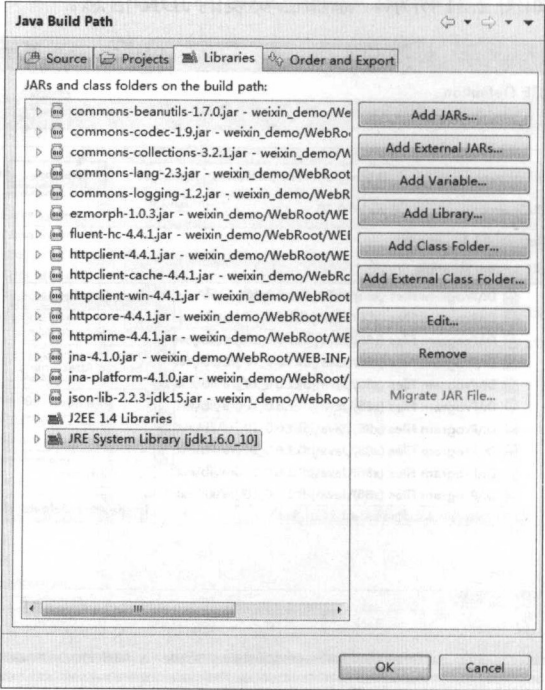


图 2.23 修改源程序 JDK

03 程序编译环境调整

在创建的项目工程上，单击“Properties”，然后单击“Java Compiler”，显示如图 2.24 所示页面。

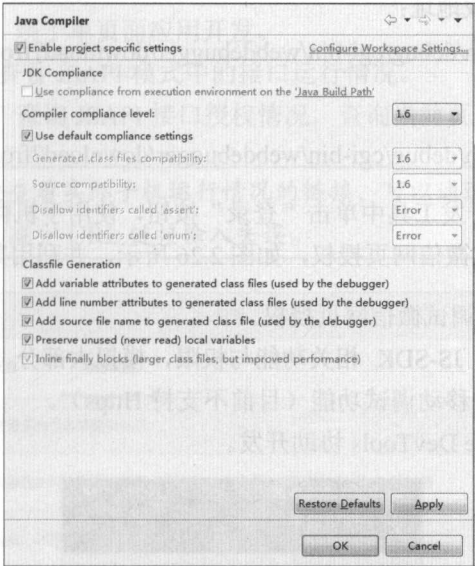


图 2.24 修改编译环境

2.2.4 微信 web 开发工具

在开发基于微信的网页授权功能时，读者通常需要在微信中输入 URL 进行开发和调试工作。由于手机的诸多限制，因此在操作上会带来很多不便，微信为帮助开发者更方便、更安全地开发和调试基于微信的网页，推出了 web 开发者工具。

微信 web 开发工具是一个桌面应用，通过模拟微信客户端的表现，使得开发者可以使用这个工具方便地进行开发和调试工作，如图 2.25 所示。

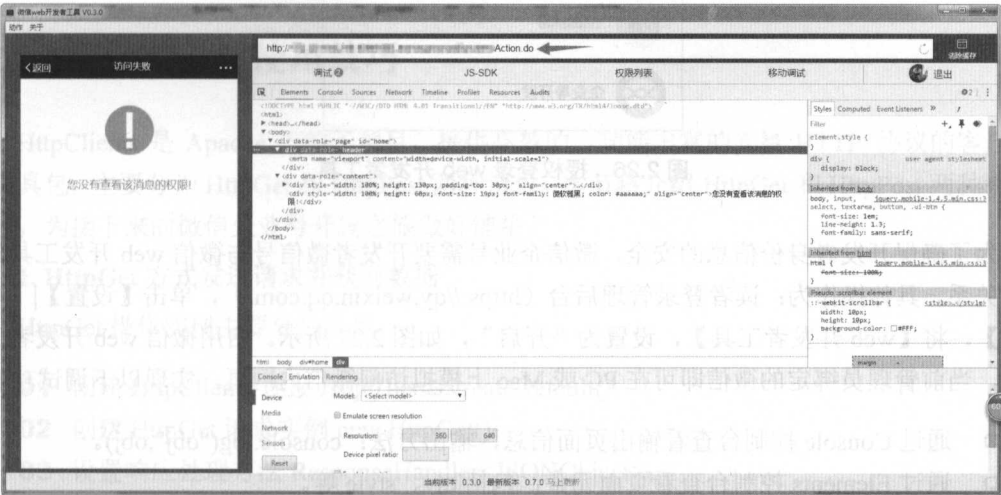


图 2.25 修改编译环境

Windows 64 位版本下载地址:

<https://mp.weixin.qq.com/debug/cgi-bin/webdebugger/download?from=mpwiki&os=x64>

Windows 32 位版本下载地址:

<https://mp.weixin.qq.com/debug/cgi-bin/webdebugger/download?from=mpwiki&os=x86>

Mac 版本下载地址:

<https://mp.weixin.qq.com/debug/cgi-bin/webdebugger/download?from=mpwiki&os=darwin>

读者可以在微信 web 开发工具中单击“登录”按钮,使用手机微信扫码登录,从而使用真实的用户身份来开发和调试微信网页授权,如图 2.26 所示,并利用其可以完成以下操作:

- 使用登录的微信号调试微信网页授权。
- 调试、检验页面的 JS-SDK 相关功能与权限,模拟大部分 SDK 的输入和输出。
- 使用基于 weinre 的移动调试功能(目前不支持 Https)。
- 利用集成的 Chrome DevTools 协助开发。

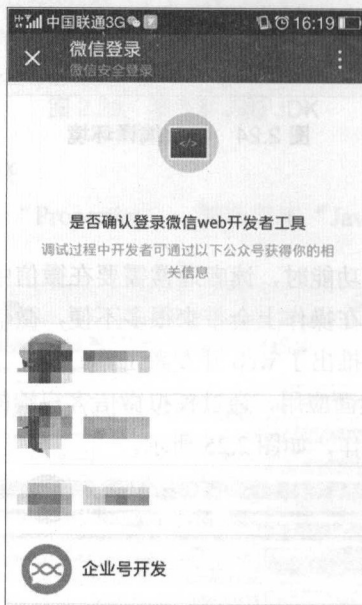


图 2.26 授权登录 web 开发者工具

为了确保开发者身份信息的安全,微信企业号需要开发者微信号与微信 web 开发工具建立绑定关系。具体操作为:读者登录管理后台(<https://qy.weixin.qq.com/>),单击【设置】|【功能设置】,将【web 开发者工具】,设置为“开启”,如图 2.27 所示。启用微信 web 开发者工具之后,当前管理员绑定的微信即可在 PC 或 Mac 上模拟访问微信内网页,实现以下调试功能:

- 通过 Console 控制台查看输出页面信息,输出方法: `console.log("obj",obj)`。
- 通过 Elements 控制台查看页面元素,包括 div、style 等。
- 通过 Sources 查看访问资源,如 JS、图片、HTML、JSP 等文件资源。

- 通过 Network 浏览请求服务列表，查询请求参数及返回结果，用于调试微信各类接口，查询接口状况。
- 通过 Resources 查看应用数据存储情况，如查看 HTML 5 特性 Local Storage、Web SQL 等本地缓存数据，用于单页面应用开发。
- 通过“JS-SDK”查看 JSAPI 模式中的接口运行情况。
- 通过“权限列表”查询 JSAPI 接口授权情况，查询当前页面具有哪些 JS 权限。



使用技巧：对于需要查看实际手机运行情况的链接，可以通过【QQ】|【我的设备】，向手机发送调试链接，减少 URL 链接输入失误。

设置		
基本信息	注册信息	功能设置
权限管理	创建并管理所有分级管理员帐号	
登录授权	设置登录授权的可信域名，登录授权时会校验请求来源和跳转目标URL的域名	
成员身份验证	可设置企业二次验证，关注验证方式以及审批加入企业号等功能	
通讯录	设置通讯录编辑功能、分级管理员可查看成员字段	
企业名片	开启后可在微信端拥有企业介绍	已开启
图片水印	微信会对发出的图文消息和图片信息使用企业名称作为水印 图片水印效果预览	已开启
企业号搜索	成员可在微信上通过搜索“企业号+企业号全称”找到该企业号（需等待一段时间后生效）	已开启
web开发者工具	开启后，当前管理员绑定的微信可在PC或Mac上模拟访问微信内网页，帮助开发者更方便地进行开发和调试。详情请查看 web开发者工具文档	

图 2.27 开启 web 开发者工具

2.3 HttpClients 使用技巧

HttpClient 是 Apache 下的子项目，提供高效的、功能丰富的支持 HTTP 协议的客户端编程工具包，主要分为 HttpGet 和 HttpPost 两种请求。本节将介绍 HttpGet 和 HttpPost 两种请求的实现，为接下来的微信企业号开发之旅做好铺垫。

1. HttpGet 方式发送请求并获取数据

HttpGet 操作实现主要分为 6 步：

- 01 创建 HttpClient 连接 HttpClient.createDefault();
- 02 创建 HttpGet 请求实例 new HttpGet();
- 03 设置响应处理方法 ResponseHandler<JSONObject>;
- 04 执行请求 httpClient.execute(httpget, responseHandler);

05 处理消息体;

06 关闭连接 httpclient.close()。

完整代码如下所示:

```
//创建链接
CloseableHttpClient httpclient = HttpClients.createDefault();
try {
    //创建 httpGet 实例
    HttpGet httpget = new HttpGet("https://qyapi.weixin.qq.com/cgi-bin/gettoken?corpid="+corpid+"&corpsecret="+corpsecret);
    //设置响应处理
    ResponseHandler<JSONObject> responseHandler = new ResponseHandler<JSONObject>() {
        public JSONObject handleResponse(
            final HttpResponse response) throws ClientProtocolException,
            IOException {
            //返回状态
            int status = response.getStatusLine().getStatusCode();
            //判断状态是否异常
            if (status >= 200 && status < 300) {
                HttpEntity entity = response.getEntity();
                if (null!=entity){
                    String result= EntityUtils.toString(entity);
                    //根据字符串生成 JSON 对象
                    JSONObject resultObj = JSONObject.fromObject(result);
                    return resultObj;
                }else{
                    return null;
                }
            } else {
                throw new ClientProtocolException("Unexpected response status:
" + status);
            }
        }
    };
    //执行请求, 返回的 JSON 对象
    JSONObject responseBody = httpclient.execute(httpget, responseHandler);
    String token="";
    if (null!=responseBody){
        token= (String) responseBody.get("access_token");//返回 token
    }
    //System.out.println("-----");
    //System.out.println("access_token:"+responseBody.get("access_token"));
    //System.out.println("expires_in:"+responseBody.get("expires_in"));
    httpclient.close();
    return token;
}
```

```

} catch (Exception e) {
    e.printStackTrace();
    return "";
}

```



使用技巧：HttpClients 连接的创建可以使用默认构造 `CloseableHttpClient httpClient = HttpClients.createDefault()`，大部分情况下，HttpClients 默认的构造函数已足够使用。

2. HttpPost 方式发送请求并获取数据

HttpPost 操作实现主要分为 8 步：

- 01** 创建 HttpClients 连接 `HttpClients.createDefault()`;
- 02** 创建请求数据实体，如 JSON 格式请求实体 `StringEntity myEntity = new StringEntity(jsonContext, ContentType.create("text/plain", "UTF-8"))`;
- 03** 创建 HttpPost 请求实例 `new HttpPost()`;
- 04** HttpPost 绑定数据请求实体 `httpPost.setEntity(myEntity)`;
- 05** 设置响应处理方法 `new ResponseHandler<JSONObject>()`;
- 06** 执行请求 `httpClient.execute(httpPost, responseHandler)`;
- 07** 处理消息体；
- 08** 关闭连接 `httpClient.close()`。

完整代码如下所示：

```

//创建一个 httpClient 链接
CloseableHttpClient httpClient = HttpClients.createDefault();
//需要访问的链接
String url=" https://qyapi.weixin.qq.com/cgi-bin/message/send?access_token=";
//新建一个 Post 请求
HttpPost httpPost= new HttpPost(url);
//发送 JSON 格式的数据
StringEntity myEntity = new StringEntity(jsonContext,
    ContentType.create("text/plain", "UTF-8"));
//设置需要传递的数据
httpPost.setEntity(myEntity);
// 创建 response 相应事件
ResponseHandler<JSONObject> responseHandler = new ResponseHandler
<JSONObject>() {
    //对访问结果进行处理
    public JSONObject handleResponse(
        final HttpResponse response) throws ClientProtocolException,
        IOException {
        //返回状态
        int status = response.getStatusLine().getStatusCode();
        //返回结果是否异常

```

```

        if (status >= 200 && status < 300) {
            HttpEntity entity = response.getEntity();
            //返回结果是否有数据
            if(null!=entity){
                String result= EntityUtils.toString(entity);
                //根据字符串生成 JSON 对象
                JSONObject resultObj = JSONObject.fromObject(result);
                return resultObj;
            }else{
                return null;
            }
        } else {
            //异常抛出异常信息, 使用 try...catch...捕获
            throw new ClientProtocolException("Unexpected response status: "
+ status);
        }
    }
};
//执行请求, 返回 JSON 对象
JSONObject responseBody = httpClient.execute(httpPost, responseHandler);
//输出对象信息
System.out.println(responseBody);
//获得 JSON 中的数据
int result= (Integer) responseBody.get("errcode");
if(0==result){
    flag=true;
}else{
    flag=false;
}
//关闭链接
httpClient.close();

```



备注: ResponseHandler 做 URL 请求的响应处理, 由 ORB 在调用期间提供给 servant, 允许 servant 稍后检索用来返回调用结果的 OutputStream。

2.4 HttpURLConnection 使用技巧

HttpURLConnection 是 URLConnection 的子类, 位于 java.net 包中, 提供一些特定于 HTTP 协议的附加功能, 允许方便地访问网络上的资源。2.3 节我们介绍了 HttpClient 的使用, 本节将介绍 HttpURLConnection 如何传递数据以及如何获取数据。

01 通过 HttpURLConnection 传递 JSON 等数据信息, 示例代码如下:

```

import java.io.File;
import java.io.FileOutputStream;

```

```

import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;
/**
 * HttpURLConnection 的使用
 */
public class TestMain {
    public static void main(String[] args) {
        String
jsonContext="{\"accessFileId\": \"QUBS0z4o-cLtnHbKh_6PmVY3kZ1wLri0RgLgBqHiJ02
5-lbER8TLb4WNrL4SV0hJ\"}";
        try {
            URL urlObj = new URL("http://***/webservice.slt?action=
getImageFromMediaId");
            HttpURLConnection con = (HttpURLConnection) urlObj.openConnection();
            //Http 正文内需要设为 true, 默认情况下是 false;
            con.setDoOutput(true);
            //设置是否从 httpURLConnection 读入, 默认情况下是 true;
            con.setDoInput(true);
            //以 POST 方式提交表单, 默认 GET 方式
            con.setRequestMethod("POST");
            con.setUseCaches(false); // post 方式不能使用缓存
            //设置请求头信息
            //con.setRequestProperty("Connection", "Keep-Alive");
            con.setRequestProperty("Charset", "UTF-8");
            //必须加数据请求格式
            con.setRequestProperty("Content-Type",
                "text/json");
            OutputStream out = con.getOutputStream();
            //向输出流中写入信息
            out.write(jsonContext.getBytes());
            //清空输出流并关闭
            out.flush();
            out.close();
            //获得输入流
            InputStream in = con.getInputStream();
            //输入流信息如果是图片
            OutputStream outputStream = new FileOutputStream(new File("G:\\app\\
aaa.jpg"));
            //添加一个头部文件
            byte[] bytes = new byte[1024];
            int cnt=0;
            while ((cnt=in.read(bytes,0,bytes.length)) != -1) {
                outputStream.write(bytes, 0, cnt);
            }

```



```
        outputStream.flush();
        outputStream.close();
        in.close();
        System.out.println("-----"+con.getContentType());

    } catch (Exception e) {
        // TODO: handle exception
    }

}

}
```



注意：数据请求必须增加数据格式，否则将无法获取数据，如传递 JSON 格式数据可以写成：`con.setRequestProperty("Content-Type","text/json")`。

URLConnection 获取传递数据也是使用流的形式，代码如下：

```
// 从输入流读取内容
BufferedReader br = new BufferedReader(new InputStreamReader
(req.getInputStream()));
String line = null;
StringBuilder sb = new StringBuilder();
while((line = br.readLine())!=null){
    b.append(line);
}
//调整编码格式
//String content=URLDecoder.decode(sb.toString(), HTTP.ISO_8859_1);
//获得数据传递内容
String content=sb.toString();
```



使用技巧：对于 JSON 格式数据，可以使用 json 包（`net.sf.json.JSONObject`）提供的 `JSONObject.fromObject()` 方法。

02 通过 HttpURLConnection 操作文件，示例代码如下：

```
/**
 * 第一部分
 */
URL urlObj = new URL("https://qyapi.weixin.qq.com/cgi-bin/media/upload?
access_token="+ token
    + "&type=file");
URLConnection con = (URLConnection) urlObj.openConnection();
con.setRequestMethod("POST"); //以 POST 方式提交表单，默认 GET 方式
con.setDoInput(true);
con.setDoOutput(true);
con.setUseCaches(false); //POST 方式不能使用缓存
```

```

//设置请求头信息
con.setRequestProperty("Connection", "Keep-Alive");
con.setRequestProperty("Charset", "UTF-8");
//设置边界
String BOUNDARY = "-----" + System.currentTimeMillis();
con.setRequestProperty("Content-Type", "multipart/form-data; boundary="+
BOUNDARY);
//请求正文信息
//第一部分:
StringBuilder sb = new StringBuilder();
sb.append("--"); // 必须多两道线
sb.append(BOUNDARY);
sb.append("\r\n");
sb.append("Content-Disposition: form-data;name=\"media\";filename=\""+
"info.csv" + "\"\r\n");
sb.append("Content-Type:application/octet-stream\r\n\r\n");
byte[] head = sb.toString().getBytes("utf-8");
//获得输出流
OutputStream out = new DataOutputStream(con.getOutputStream());
//输出表头
out.write(head);
//文件正文部分
//把文件以流文件的方式推入到 URL 中
DataInputStream in = new DataInputStream(new ByteArrayInputStream
(content.getBytes()));
int bytes = 0;
byte[] bufferOut = new byte[1024];
while ((bytes = in.read(bufferOut)) != -1) {
    out.write(bufferOut, 0, bytes);
}
in.close();
//结尾部分
byte[] foot = ("\r\n--" + BOUNDARY + "--\r\n").getBytes("utf-8");
//定义最后数据分隔线
out.write(foot);
out.flush();
out.close();
StringBuffer buffer = new StringBuffer();
BufferedReader reader = null;
try {
    //定义 BufferedReader 输入流来读取 URL 的响应
    reader = new BufferedReader(new InputStreamReader(con.getInputStream()));
    String line = null;
    while ((line = reader.readLine()) != null) {
        //System.out.println(line);
        buffer.append(line);
    }
}

```



```

    }
    if(result==null){
        result = buffer.toString();
    }
} catch (IOException e) {
    System.out.println("发送 POST 请求出现异常! " + e);
    e.printStackTrace();
    throw new IOException("数据读取异常");
} finally {
    if(reader!=null){
        reader.close();
    }
}
}

```



使用技巧: `con.getContentType()` 可用于获取请求的数据格式。

2.5 Properties 配置文件使用

Properties 文件是 Java 中的一种配置文件，类似于 XML 文件等都属于配置文件，与 XML 文件不同，Properties 内容以键值对的形式表示，通过 Properties 文件的方式方便程序部署时进行数据参数修改。

在项目产品化中，Properties 也是关键的一部分，项目产品化即同一程序不同情况部署，微信企业号的开发也比较适合产品化。产品化中的个性化差异可以通过数据库配置、XML 配置、Properties 文件配置等方式解决，有兴趣的读者可以详细了解，这里只介绍其中的一种方式——Properties 文件的配置读取。

在工程目录 `src` 目录下，创建 Properties 文件，命名为“`connection.properties`”，并添加测试数据（这里向读者演示的是数据库连接信息的获取，当然，也可以采用 Hibernate 在 XML 中配置，这里只是做文件操作演示），Properties 文件内容如下：

```

driver=oracle.jdbc.driver.OracleDriver
url=jdbc:oracle:thin:@localhost:1521:orcl
user=shop
pwd=123456

```

接下来采用静态块的方式，获取 Properties 文件信息：

```

package com.myfactory;
import java.util.ResourceBundle;
public class Myfactory {
    public static String driver;
    public static String url;
    public static String user;
    public static String pwd;
    //静态块

```

```

static{
    //方法一
    ResourceBundle rsb = ResourceBundle.getBundle("connection");
    driver = rsb.getString("driver");
    url = rsb.getString("url");
    user = rsb.getString("user");
    pwd = rsb.getString("pwd");
    //方法二
    //Properties properties = new Properties();
    //InputStream in = WxUtil.class.getClassLoader().
    //    getResourceAsStream("com/service/wxConfig.properties");
    //properties.load(in);
    //messageCorpId = properties.get("corpId")+"";
}
}

```



备注：静态块在类加载时将优先执行。需要注意的是，并不是所有的信息都需要进行 Properties 等文件配置，部分信息可以使用 static final 的形式进行定义，而且配置程度越高的产品越需要进行证书、加密、混淆等操作，防止盗卖等现象。

2.6 接口调试工具

微信企业号接口调试工具，是微信企业号开发的又一力作，它能够帮助读者检测调用 API 时发送的请求参数是否正确，是否能够获得服务器的验证结果。接口调试工具访问地址：<http://qydev.weixin.qq.com/debug>，详细步骤如下。

- 01** 选择需要调试的接口类型，包括建立连接、管理素材文件等。
- 02** 选择接口列表，确定需要调试的接口，并填写接口信息（红色星号表示该字段为必填项），如图 2.28 所示。

图 2.28 选择接口列表

03 填写完成之后，单击“检查问题”按钮，如果请求数据无误，将返回结果信息，如图 2.29 所示，如果请求数据有误，将返回错误说明，如图 2.30 所示。

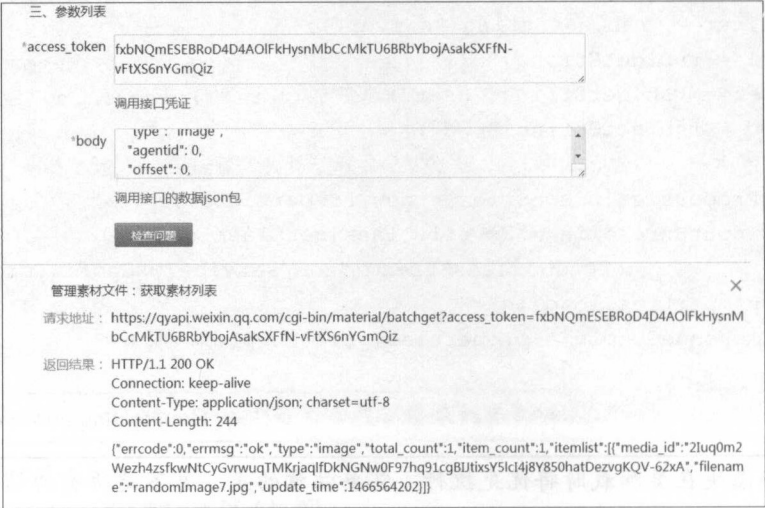


图 2.29 接口调试正确返回结果信息

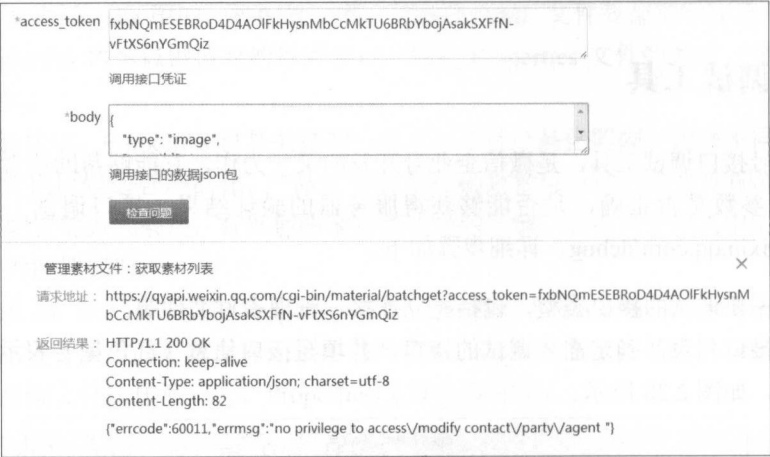


图 2.30 接口调试错误返回错误说明

2.7 发布外网服务

既然微信企业号开发是一个特殊的 B/S 手机网站，那么外网域名则是必需的，读者企业号服务需要通过外网域名才能够发送微信接口指令，同样，只有通过外网域名微信企业号才能够推送客户操作信息，连接开发的项目功能及页面，实现读者服务与微信企业号之间的互动操作及信息传递，从而使客户能正常使用微信企业号。

本节利用内网版“花生壳”，简单演示如何利用“花生壳”发布外网域名有条件的读者可以直接向公司申请二级域名（在备案的 ICP 域名下增加二级域名），而没有域名的读者，则可以申请“花生壳”等服务商，将获得免费的外网域名。打开“花生壳”之后，双击申请的域名，

打开域名配置页面，如图 2.31 所示，填写“应用名称”、“内网主机”和“端口号”。

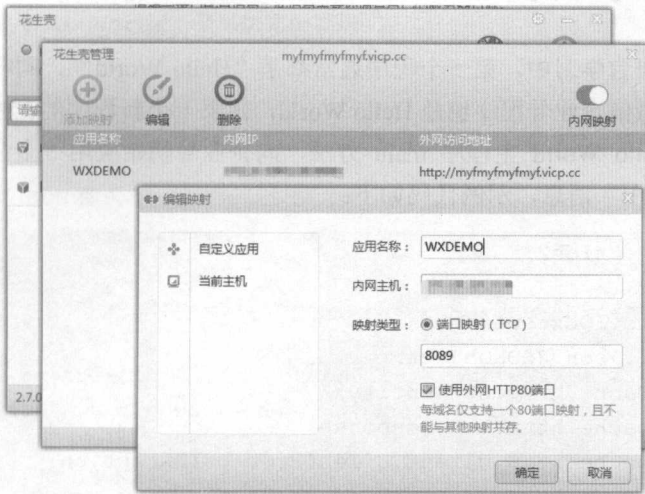


图 2.31 发布外网服务



建议：可以使用客户或者自己企业的域名，因为在 JSAPI 模式中，必须使用 ICP 备案的域名，个人新申请的域名未在 ICP 备案。

2.8 公众平台消息模式

了解完公众平台及开发基础之后，将正式进入微信企业号开发，不过在这之前，首先需要了解企业号的消息模式，进行模块化的学习，从而使学习结构更清晰。在企业号开发中，公众平台消息模式主要分为以下几种：

- 主动调用模式，是政府、企业等通过企业号向员工发送信息的模式，主要实现信息的推送功能。
- 被动回调模式，是员工向企业号发送信息以及触发事件，而产生消息的模式，包括消息的接收、被动响应以及主动推送等。
- 基于微信 JS-SDK 的 JSAPI 模式，是通过微信内置浏览器，结合微信 JS-SDK 开发出能与微信互动的 B/S 手机网站的模式。
- 企业会话模式，是微信最近推出的与企业自有的 IM（Instant Messaging，即时通信）软件进行消息对接的模式。

企业会话模式在开发上可能比较少，多以直接使用为主，而主动调用模式、被动回调模式以及基于微信 JS-SDK 的 JSAPI 模式在微信开发中是较为常用的模式，在订阅号、服务号开发中这三种模式也是必不可少的一部分，在后面的章节中将对微信企业号开发的四种消息模式以及数据的安全访问进行详细解释说明。

备注：被动回调模式下，可以使用主动调用模式下的所有推送接口；反之、则不行。

2.9 微信企业号入门 Hello World

在计算机编程语言学习中，第一个程序通常都是“Hello World”，同样，在微信企业号开发中，我们第一个微信企业号程序也是 Hello World，代表一个新程序的诞生，向世界问候。下面就来创建一个 Hello World 主程序 main 方法，向企业号初始应用“企业小助手”发送一条“XXX,Hello World!!!”消息，示例代码如下：

```
package myf.caption2;

import java.io.IOException;
import net.sf.json.JSONObject;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.ResponseHandler;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.ContentType;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

public class HelloWorld {
    //主程序
    public static void main(String[] args) {
        HelloWorld helloWorld= new HelloWorld();
        String jsonContext="{
            +\"\\\"touser\\\": \\\"muyunfei\\\", \"
            +\"\\\"toparty\\\": \\\"\\\", \"
            +\"\\\"totag\\\": \\\"\\\", \"
            +\"\\\"msgtype\\\": \\\"text\\\", \"
            +\"\\\"agentid\\\": 0, \"
            +\"\\\"text\\\": {
                +\"\\\"content\\\": \\\"XXX,Hello World!!!\\\"\"
            +\"}, \"
            +\"\\\"safe\\\": 0\"
        }\"";

        String corpId = "xxxxxxxxxx";
        String corpsecret = "xxxxxxxxxxxxxxxxxx";
        helloWord.sendReqMsg(jsonContext, corpId, corpsecret);
    }

    /**
     * 发送消息
    */
}
```

```

    * @param jsonContext JSON 字符串
    * @param corpId 微信企业号标识
    * @param corpsecret 管理组凭证密钥
    * @return
    */
    public JSONObject sendReqMsg(String jsonContext,String corpId,String
corpsecret){
        //消息 JSON 格式
        JSONObject result =null;
        //获得 token
        String token=getTokenFromWx(corpId, corpsecret);
        try {
            CloseableHttpClient httpClient = HttpClient.createDefault();
            HttpPost httpPost= new HttpPost("https://qyapi.weixin.qq.com" +
                "/cgi-bin/message/send?access_token="+token);
            //发送 JSON 格式的数据
            StringEntity myEntity = new StringEntity(jsonContext,
                ContentType.create("text/plain", "UTF-8"));
            //设置需要传递的数据
            httpPost.setEntity(myEntity);
            // Create a custom response handler
            ResponseHandler<JSONObject> responseHandler = new ResponseHandler
<JSONObject>() {
                //对访问结果进行处理
                public JSONObject handleResponse(final HttpResponse response)
throws ClientProtocolException, IOException {
                    int status = response.getStatusLine().getStatusCode();
                    if (status >= 200 && status < 300) {
                        HttpEntity entity = response.getEntity();
                        if(null!=entity){
                            String result= EntityUtils.toString(entity);
                            //根据字符串生成 JSON 对象
                            JSONObject resultObj = JSONObject.fromObject
(result);

                            return resultObj;
                        }else{
                            return null;
                        }
                    } else {
                        throw new ClientProtocolException("Unexpected response
status: " + status);
                    }
                }
            };
            //返回的 JSON 对象

```



```

        JSONObject responseBody = httpClient.execute(httpPost, responseHandler);
        System.out.println(responseBody.toString());
        result=responseBody;
        httpClient.close();
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return result;
}
/**
 * 获取 Token
 * @param corpId
 * @param corpsecret
 * @return
 */
public String getTokenFromWx(String corpId,String corpsecret){
    //获取的标识
    String token="";
    CloseableHttpClient httpClient = HttpClients.createDefault();
    try {
        //利用 GET 形式获得 Token
        HttpGet httpget = new HttpGet("https://qyapi.weixin.qq.com/cgi-bin/gettoken?corpid="+corpId+"&corpsecret="+corpsecret);
        // Create a custom response handler
        ResponseHandler<JSONObject> responseHandler = new ResponseHandler<JSONObject>() {
            public JSONObject handleResponse(
                final HttpResponse response) throws ClientProtocolException,
                IOException {
                int status = response.getStatusLine().getStatusCode();
                if (status >= 200 && status < 300) {
                    HttpEntity entity = response.getEntity();
                    if(null!=entity){
                        String result= EntityUtils.toString(entity);
                        //根据字符串生成 JSON 对象
                        JSONObject resultObj = JSONObject.fromObject
(result);

                        return resultObj;
                    }else{
                        return null;
                    }
                } else {
                    throw new ClientProtocolException("Unexpected response
status: " + status);
                }
            }
        };
    }

```

```

    }
};
//返回的 JSON 对象
JSONObject responseBody = httpClient.execute(httpget, responseHandler);
if (null!=responseBody) {
    token= (String) responseBody.get("access_token");//返回 token
}
httpClient.close();
} catch (Exception e) {
    e.printStackTrace();
}
return token;
}
}

```

执行主函数，效果如图 2.32 所示。



图 2.32 Hello World 入门

备注：agentid 为企业号应用 ID，默认应用“企业小助手”agentid 为 0，且不可删除。CorpID 与 CorpSecret 为微信企业号标识和管理组密钥，可通过微信管理端获取，详细获取信息请参照 1.2.7 节。

【第二篇】

微信企业号开发核心技术

- 第 3 章 主动调用模式
- 第 4 章 被动回调模式
- 第 5 章 JSAPI 模式
- 第 6 章 企业会话模式
- 第 7 章 通讯录管理及异步任务
- 第 8 章 数据安全访问策略
- 第 9 章 数据库及服务中间件

第3章

主动调用模式

通过对前面两章的学习，我们已经了解微信开发的基础知识，本章将深入学习微信企业号开发中的主动调用模式。主动调用模式是企业号开发常用的模式之一，主要功能是发送微信各类消息，通过对本章的学习，使读者掌握主动调用模式的消息发送方式，掌握各类消息的消息结构，以及掌握如何进行数据缓存处理等。

本章主要涉及的知识点如下。

- 什么是主动调用模式：掌握主动调用模式的基础知识。
- 关键数据 Token：学会申请 AccessToken，以及对 AccessToken 数据进行缓存处理。
- 各类消息体：掌握各类消息的信息结构，学会发送各类消息。
- 素材管理：了解微信对多媒体文件的处理方式，学会上传、删除、修改、查看临时/永久素材文件。
- 企业号应用管理：学会通过接口控制企业号各应用。
- 业务与接口的实际应用：通过本章最后的示例，学习主动调用模式下接口的使用方法，通过本章讲解的知识，正确调用微信接口。

3.1 主动调用模式介绍

企业应用调用企业号提供的接口，管理或查询企业号后台所管理的资源，或给成员发送消息等，称为主动调用模式。简单地说，由企业号向员工推送消息，即为主动调用模式。主动调用模式是企业号开发中最基本的连接模式，主动调用模式的特点如下。

- 访问协议：采用 HTTPS 协议的方式。
- 数据格式：JSON 数据格式。
- 数据编码方式：UTF-8 编码。
- 访问域名：<https://qyapi.weixin.qq.com>。
- 加密方式：数据包不需要加密。
- 在每次主动调用企业号接口时需要带上 AccessToken 参数，AccessToken 的详细说明将在 3.2 节中介绍。

主动调用模式是微信企业级轻应用中广泛应用的模式，可以利用其开发企业内部资讯的实时推送、业务派单、通知公告、微刊推送等各种提醒信息，提高消息的时效性；也可以推送企业相册、员工关怀信息等，促进企业文化发展；同时，还可以推送问卷调查、投票等辅助决策信息，辅助公司决策。总之，主动调用模式是企业号开发中非常重要的环节，也是必不可少的环节。

3.2 申请 AccessToken

AccessToken 是企业号接口调用的全局唯一票据，是每次主动调用企业号接口时必须携带的参数。

AccessToken 参数的获取由 CorpID 和 Secret 调用接口获得。CorpID 是企业号的标识，每个企业号拥有唯一的 CorpID。Secret 是管理组凭证密钥，在创建管理员“管理组”时生成，系统管理员可通过管理端的权限管理功能创建“管理组”，分配“管理组”相应的应用访问权限，完成后，管理组即可获得唯一的 Secret，系统管理员可通过权限管理查看所有管理组的 Secret，其他管理员可通过设置中的开发者凭据查看（操作步骤在 1.2.7 节中已经介绍），不同的 Secret 将获得不同的 AccessToken。

当企业应用调用企业号接口时，企业号后台会根据此次访问的 AccessToken、校验访问的合法性以及所对应的管理组的管理权限返回相应的结果。

注意：分配的管理员“管理组”权限不宜过大，这样可减少操作失误，提高信息的安全性。

企业号申请 AccessToken 票据接口详细说明如下。

- Https 请求链接如下：
https://qyapi.weixin.qq.com/cgi-bin/gettoken?corpid=id&corpsecret=secret
- 数据请求方式：
GET 方式进行数据请求。
- 参数说明：详细说明如表 3.1 所示。

表 3.1 申请AccessToken请求参数说明

参数	是否必需	说明
corpid	是	企业号唯一标识CorpID
corpsecret	是	管理组凭证密钥Secret

- 权限说明：
每个 Secret 代表了对应用、通讯录的不同权限，不同的管理组拥有不同的 Secret。
- 执行结果说明
 - 执行正确时返回的正确的 JSON 结果：

```
{
  "access_token": "accesstoken000001",
```

```
"expires_in": 7200
}
```

详细说明如表 3.2 所示。

表 3.2 执行正确，返回JSON数据说明

参数	说明
access_token	获取到的凭证，长度为64~512字节
expires_in	凭证的有效时间（秒）

- 返回的失败的 JSON 结果：

```
{
  "errcode": 43003,
  "errmsg": "require https"
}
```

详细说明如表 3.3 所示。

表 3.3 返回错误JSON数据说明

参数	说明
errcode	返回错误码，详见附录B
errmsg	返回错误说明

完整示例代码如下（微信企业号开发相关 jar 包在 2.1.3 节中下载）：

```
package com.myf;

import java.io.IOException;
import net.sf.json.JSONObject;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.ResponseHandler;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

public class GetTokenMain {

    public static void main(String[] args) {
        String token = GetTokenMain.getToken("输入 corpID", "输入管理组密钥");
        System.out.println("-----");
        System.out.println("获得的 Token 为:"+token);
        System.out.println("-----");
    }
}
```



```

//获取微信 Token
public static String getToken(String corpid,String corpsecret){
    CloseableHttpClient httpclient = HttpClients.createDefault();
    try {
        //利用 GET 形式获得 Token
        HttpGet httpget = new HttpGet("https://qyapi.weixin.qq.com/cgi-bin/gettoken?corpid="+corpid+"&corpsecret="+corpsecret);
        // Create a custom response handler
        ResponseHandler<JSONObject> responseHandler = new ResponseHandler<JSONObject>() {

            public JSONObject handleResponse(
                final HttpResponse response) throws ClientProtocolException,
                IOException {

                int status = response.getStatusLine().getStatusCode();
                if (status >= 200 && status < 300) {
                    HttpEntity entity = response.getEntity();
                    if(null!=entity){
                        String result= EntityUtils.toString(entity);
                        //根据字符串生成 JSON 对象
                        JSONObject resultObj = JSONObject.fromObject
(result);

                        return resultObj;
                    }else{
                        return null;
                    }
                } else {
                    throw new ClientProtocolException("Unexpected response
status: " + status);
                }
            }

        };
        //返回的 JSON 对象
        JSONObject responseBody = httpclient.execute(httpget, responseHandler);
        String token="";
        if(null!=responseBody){
            token= (String) responseBody.get("access_token");//返回 token
        }
        //System.out.println("-----");
        //System.out.println("access_token:"+responseBody.get("access_
token"));
        //System.out.println("expires_in:"+responseBody.get("expires_in"));

        httpclient.close();
    }
}

```

```
        return token;
    } catch (Exception e) {
        e.printStackTrace();
        return "";
    }
}
```

输出信息如图 3.1 所示。

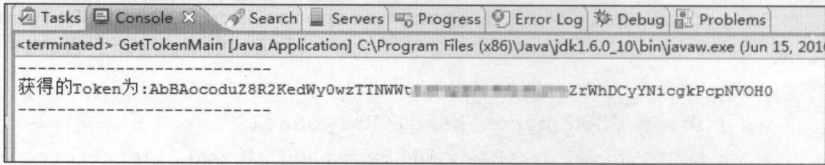


图 3.1 输出 Token 信息



备注：access_token 需要至少保留 512 字节的存储空间，在 Java 中使用 String 即可，因为 String 的 count 为 int 类型，所以完全不用担心存储空间的问题。

3.3 AccessToken 的缓存处理

AccessToken 是主动模式下重要的接口票据，利用 AccessToken 便可以调用企业号后台提供的各种接口，向关注人员发送各类信息，查看、管理企业号各类资源，并且能够管理企业各个应用。权限越大，安全性就更加需要防范，为了提高企业号的安全性，AccessToken 在默认情况下具有一定的调用频率限制和有效期。对于超过调用频率的企业号，再次发起 AccessToken 票据申请，企业号将返回错误提示代码。对于已经超过有效期的 Token，也无法正常使用企业号各接口，并返回错误提示。正常情况下，AccessToken 的有效期为 7200 秒（目前是 7200 秒，需要以返回结果中的 expires_in 为准），有效期内重复获取将获得相同的结果。在开发企业号过程中，AccessToken 必须进行缓存处理。



注意：早期版本的 AccessToken 支持有效期内续期，但于 2016 年 3 月 11 日取消自动续期功能。

本节将向读者演示如何对 AccessToken 进行缓存处理，可能有一部分读者通过网络下载了 AccessToken 缓存处理包，其实利用简单的几行代码即可完成一个缓存操作。本节我们讲解的是通过“static”静态全局变量，完成 AccessToken 数据缓存。

01 定义三个静态的全局变量 access_token（获得票据）、access_token_date（获得票据的时间）以及 accessTokenInvalidTime（票据的有效时间，默认 7200 秒），示例代码如下：

```
//主动调用：发送消息 AccessToken token
public static String access_token;
```

```
//主动调用：请求 Token 的时间
public static Date access_token_date;
//Token 有效时间，默认 7200 秒，每次请求更新，用于判断 Token 是否超时
public static long accessTokenInvalidTime=7200L;
```

02 处理过程中，优先判断 access_token 是否为空，为空则直接向企业号申请 access_token；如果不为空，则借助 access_token_date(申请时间)判断 access_token 是否在有效时间内。若在有效时间内，则直接返回缓存的 access_token，否则向企业号申请新的 access_token，并保存 access_token_date 和 access_token，详细处理流程如图 3.2 所示。

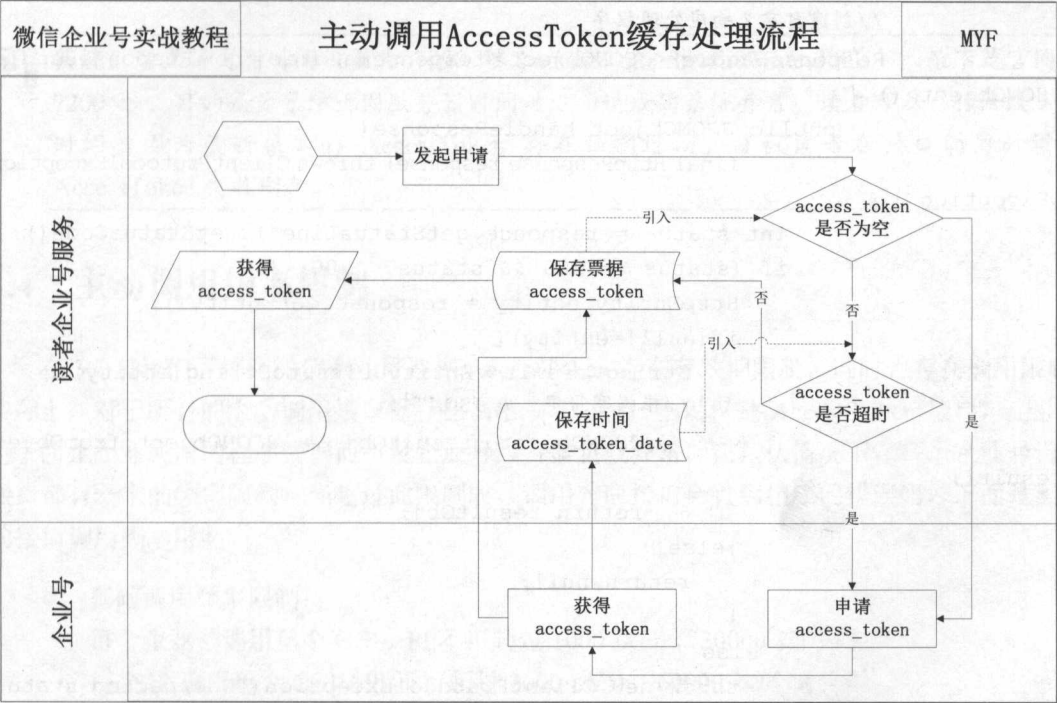


图 3.2 AccessToken 缓存处理

完整示例代码如下所示：

```
/**
 * 从微信获得 access_token
 * @return
 */
public String getTokenFromWx(){
    //微信企业号标识
    String corpid="*****"; //请读者改成自己的 CorpID
    //管理组凭证密钥
    String corpsecret="*****"; //请读者改成自己的 Secret
    //获取的标识
    String token="";
    //1.判断 access_token 是否存在，不存在则直接申请
```

```

//2.判断时间是否过期,过期(>=7200秒)申请,否则不用请求直接返回以后的Token
if(null==access_token||"".equals(access_token)|| (new
Date().getTime()-access_token_date.getTime())>=((accessTokenInvalidTime-200L
)*1000L)){
    CloseableHttpClient httpClient = HttpClients.createDefault();
    try {
        //利用 GET 形式获得 Token
        HttpGet httpget = new HttpGet("https://qyapi.weixin.qq.com/cgi-
bin/gettoken?corpid="
+corpid+"&corpsecret="+corpsecret);
        //创建自定义响应处理程序
        ResponseHandler<JSONObject> responseHandler = new ResponseHandler
<JSONObject>() {
            public JSONObject handleResponse(
                final HttpResponse response) throws ClientProtocolException,
IOException {
                int status = response.getStatusLine().getStatusCode();
                if (status >= 200 && status < 300) {
                    HttpEntity entity = response.getEntity();
                    if(null!=entity){
                        String result= EntityUtils.toString(entity);
                        //根据字符串生成 JSON 对象
                        JSONObject resultObj = JSONObject.fromObject
(result);
                        return resultObj;
                    }else{
                        return null;
                    }
                } else {
                    throw new ClientProtocolException("Unexpected status:
"+ status);
                }
            }
        };
        //返回的 JSON 对象
        JSONObject responseBody = httpClient.execute(httpget, responseHandler);
        //正确返回结果,进行更新数据
        if(null!=responseBody&&null!=responseBody.get("access_token")){
            //设置全局变量
            token= (String) responseBody.get("access_token");//返回 token
            //更新 Token 有效时间
            accessTokenInvalidTime=Long.valueOf(responseBody.get
("expires_in")+"" );
            //设置全局变量
            accessToken=token;
            accessTokenDate=new Date();

```



```

    }
    httpclient.close();
} catch (Exception e) {
    e.printStackTrace();
}
} else {
    token=access_token;
}
return token;
}

```



备注：这里将更新时间设置为 7000 秒（accessTokenInvalidTime-200L），而不是官网的 7200 秒，目的是为了防止因服务器时间延迟而造成的系统异常。读者可以自行修改刷新时间，因为最新版本的 AccessToken 存在强制过期，建议读者在接口调用时增加 AccessToken 失效刷新。

3.4 主动调用频率限制

在 3.3 节我们了解到微信企业号票据 AccessToken 存在有效期限制，同时也存在调用限制。实际上，对于所有的主动调用接口都存在调用频率限制，不仅仅是 AccessToken 接口，企业号为了防止企业应用的程序错误而引发企业号服务器负载异常，在默认情况下，每个企业号调用接口都有一定的频率限制，当超过此限制时，调用对应接口会收到相应的错误码，下面是当前的接口调用频率限制。

□ 基础调用频率限制：

每个企业号调用单个 CGI/API 不可超过 1000 次/分，30000 次/小时；

每个 IP 调用单个 CGI/API 不可超过 2000 次/分，60000 次/小时；

第三方应用每个 IP 调用单个 CGI/API 不可超过 20000 次/分，600000 次/小时。

□ 消息推送接口调用频率限制：

每个企业号信息推送数量不可超过账号上限数×30 人次/天。

□ 创建账号频率限制：

每个企业号创建账号数不可超过账号上限数×3/月。

□ 创建应用频率限制：

每个企业号最大应用数限制为 30 个，创建应用次数不可超过 30×3/月。

3.5 信息推送

信息推送是微信公众平台开发的重要功能，实现了消息的时效性，通过对本节的学习，读者将掌握如何推送各类消息，以及了解消息封装的各种方式。本节将使用继承、接口、字符串拼接、JSON 包等形式对消息实现封装推送，方便读者选择合适的封装方法。

3.5.1 接口说明

在学习信息推送之前，还是先了解下信息推送的基本信息，基本信息如下。

(1) Https 请求链接如下：


https://qyapi.weixin.qq.com/cgi-bin/message/send?access_token=ACCESS_TOKEN

(2) 数据请求方式。

POST 方式进行数据请求。

(3) 消息类型。

消息型应用支持文本、图片、语音、视频、文件、图文等消息类型。主页型应用只支持文本消息类型，且文本长度不可超过 20 个字。

 **备注：**消息型应用与主页型应用有本质区别，消息型应用是进入微信窗口并可以自定义菜单，主页型应用则直接跳转到页面。


(4) 参数说明：详细说明如表 3.4 所示。

表 3.4 信息推送说明

参数	是否必须	说明
access token	是	接口调用凭证

(5) 权限说明。

- ❑ 信息成功推送需要满足以下条件，否则信息推送失败；
- ❑ 信息接收人必须在应用的可见范围内；
- ❑ 管理组 Secret 对当前应用具有信息发送的权限；
- ❑ 信息接收人具有查看权限；
- ❑ 信息推送的 AccessToken 必须有效。

 **注意：**如果信息推送出现 {"errcode":60011,"errmsg":"no privilege to access/modify contact/party/agent "}, 则表示权限不足，需要登录管理端后台【设置】|【功能设置】|【权限管理】|【选择管理组】|【应用权限】增加信息发送权限。

(6) 执行结果说明。

如果无权限，则本次发送失败；如果收件人不存在或未关注，则仍然执行信息推送。两种情况下均返回无效的部分。执行接口推送，返回的 JSON 结果如下：

```
{
  "errcode": 0,
  "errmsg": "ok",
  "invaliduser": "UserID1",
  "invalidparty": "PartyID1",
  "invalidtag": "TagID1"
}
```


详细说明如表 3.5 所示。

表 3.5 执行正确，并返回JSON数据说明

参数	说明
errcode	执行结果，0成功，1失败
errmsg	接口执行信息
invaliduser	无效的userid
invalidparty	无效的部门
invalidtag	无效的标签



注意：返回的 userid 不区分大小写，均为小写。

完整示例代码如下：

```
/**
 * 主动发送消息
 * @param mess
 * @return
 */
public JSONObject sendReqMsg(ReqBaseMessage mess){
    //消息 JSON 格式
    String jsonContext=mess.toJsonStr();
    JSONObject result =null;
    //获得 Token
    String token=getTokenFromWx();
    boolean flag=false;
    try {
        CloseableHttpClient httpClient = HttpClients.createDefault();
        HttpPost httpPost= new HttpPost("https://qyapi.weixin.qq.com/cgi-bin/message/send?access_token="+token);
        //发送 JSON 格式的数据
        StringEntity myEntity = new StringEntity(jsonContext,
            ContentType.create("text/plain", "UTF-8"));
        //设置需要传递的数据
        httpPost.setEntity(myEntity);
        // Create a custom response handler
        ResponseHandler<JSONObject> responseHandler = new ResponseHandler<JSONObject>() {
            //对访问结果进行处理
            public JSONObject handleResponse(
                final HttpResponse response) throws ClientProtocolException,
                IOException {
                int status = response.getStatusLine().getStatusCode();
                if (status >= 200 && status < 300) {
                    HttpEntity entity = response.getEntity();
```

```
        if(null!=entity){
            String result= EntityUtils.toString(entity);
            //根据字符串生成 JSON 对象
            JSONObject resultObj = JSONObject.fromObject(result);
            return resultObj;
        }else{
            return null;
        }
    } else {
        throw new ClientProtocolException("Unexpected response
status: " + status);
    }
}

};
//返回的 JSON 对象
JSONObject responseBody = httpclient.execute(httpPost, responseHandler);
result=responseBody;
httpclient.close();
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
return result;
}
```



备注: getTokenFromWx()为 3.3 节中介绍的 AccessToken 缓存方法, ReqBaseMessage 为所有信息推送类的父类, 可以是抽象类, 也可以是接口。

3.5.2 推动文本消息

文本类型消息(也称 Text 消息), 允许信息保密("safe": "1"), 保密消息将带有水印, 并且仅能被信息接收人查看, Text 消息详细说明如下。

(1) 消息请求结构体:

```
{
    "touser": "UserID1|UserID2|UserID3",
    "toparty": " PartyID1 | PartyID2 ",
    "totag": " TagID1 | TagID2 ",
    "msgtype": "text",
    "agentid": 1,
    "text": {
        "content": "Holiday Request For Pony(http://xxxxx)"
    },
    "safe": "0"
}
```

注意：文本长度不得超过 600 字。主页型应用支持文本类型，文本长度不得超过 20 个字。

(2) 参数说明。详细说明如表 3.6 所示。

表 3.6 Text消息体参数说明

参数	是否必需	说明
touser	否	成员ID列表（消息接收者，多个接收者用‘ ’分隔，最多支持1000个）。特殊情况：若指定为@all，则向关注该企业应用的全部成员发送
toparty	否	部门ID列表，多个接收者用‘ ’分隔，最多支持100个。当touser为@all时忽略本参数
totag	否	标签ID列表，多个接收者用‘ ’分隔。当touser为@all时忽略本参数
msgtype	是	消息类型，此时固定为：text
agentid	是	企业应用的ID，整型。可在应用的设置页面查看
content	是	消息内容，最长不超过2048个字节。注意：主页型应用推送的文本消息在微信端最多只显示20个字（包含中英文）
safe	否	表示是否是保密消息，0表示否，1表示是，默认是0

注意：仅 Text 消息支持主页型应用。touser、toparty、totag 三个不能同时为空，之后章节中将不再说明。

(3) 示例代码

示例封装的方法是抽象父类+字符串拼接的方式，新建抽象类 ReqBaseMsg.java，它是所有主动调用模式下信息推送的父类。

```
package myf.caption3.demo3_5;
/**
 * 所有主动调用模式下信息推送的父类
 * @author muyunfei
 */
public abstract class ReqBaseMsg {

    /**
     * 此处为微信中添加的账号，并非个人微信号
     * 成员 ID 列表（消息接收者，多个接收者用'|'分隔，最多支持 1000 个）
     * 特殊情况：指定为@all 时，则向关注该企业应用的全部成员发送
     */
    protected String touser;

    /**
     * 部门 ID 列表，多个接收者用'|'分隔，最多支持 100 个。当 touser 为@all 时忽略本参数
     */
    protected String toparty;
```

```

    /**
     * 标签 ID 列表，多个接收者用 '|' 分隔。当 touser 为 @all 时忽略本参数
     */
    protected String totag;

    /**
     * 消息类型，文本类型为：text、image、voice、video、file、news、mapnews
     */
    protected String msgtype;

    /**
     * 企业号应用 ID
     */
    protected String agentid;

    /**
     * 定义抽象方法，所有子类必须实现
     */
    public abstract String toJsonStr();
}
新建文本消息类 WxTextMessage.java，完整代码如下：package myf.caption3.demo3_5;
/**
 * 文本消息
 * @author muyunfei
 */
public class WxTextMessage extends ReqBaseMsg {
    //有参构造下必须加无参数构造
    public WxTextMessage(){}
    //消息内容
    public String content;
    //表示是否是保密消息：0 表示否，1 表示是，默认是 0
    protected String safe;

    public String getSafe() {
        return safe;
    }

    public void setSafe(String safe) {
        this.safe = safe;
    }

    public String getContent() {
        return content;
    }

    public void setContent(String content) {
        this.content = content;
    }

    //有参构造
    public WxTextMessage(String touser, String toparty, String totag,
        String msgtype, String agentid, String content, String safe) {

```



```

super();
super.touser = touser;
super.toparty = toparty;
super.totag = totag;
super.msgtype = msgtype;
super.agentid = agentid;
this.content = content;
this.safe = safe;
}
//装换成JSON字符串
public String toJsonStr(){
    StringBuffer jsonStr=new StringBuffer("{}");
    StringBuffer str_tmp= new StringBuffer("");
    //{"touser": "@all","msgtype": "text","agentid":
    "0","text": {"content": "helloworld"},"safe": "0"}";
    if(null!=touser&&!"".equals(touser)){
        if(!"".equals(str_tmp.toString())){
            str_tmp.append(",");
        }
        str_tmp.append("\"touser\": \""+touser+"\"");
    }
    if(null!=toparty&&!"".equals(toparty)){
        if(!"".equals(str_tmp.toString())){
            str_tmp.append(",");
        }
        str_tmp.append("\"toparty\": \""+toparty+"\"");
    }
    if(null!=totag&&!"".equals(totag)){
        if(!"".equals(str_tmp.toString())){
            str_tmp.append(",");
        }
        str_tmp.append("\"totag\": \""+totag+"\"");
    }
    if(null!=msgtype&&!"".equals(msgtype)){
        //去除空格
        msgtype=msgtype.trim();
        //判断是否加逗号
        if(!"".equals(str_tmp.toString())){
            str_tmp.append(",");
        }
        str_tmp.append("\"msgtype\": \""+msgtype+"\"");
        //文本消息
        if(null!=content&&!"".equals(content)){
            if(!"".equals(str_tmp.toString())){
                str_tmp.append(",");
            }

```

```

        str_tmp.append("\"text\": {\"content\": \""+content+"\"}");
    }

}

if(null!=agentid&&!("{}", equals(agentid)){
    if!("{}", equals(str_tmp.toString())){
        str_tmp.append(",");
    }
    str_tmp.append("\"agentid\": \""+agentid+"\"");
}

if(null!=safe&&!("{}", equals(safe)){
    if!("{}", equals(str_tmp.toString())){
        str_tmp.append(",");
    }
    str_tmp.append("\"safe\": \""+safe+"\"");
}

jsonStr.append(str_tmp);
jsonStr.append("{}");
return jsonStr.toString();
}
}

```

方法执行如下:

```

//发送文本消息
String context="姓名: 牟云飞\r\n"
    +"手机: 155625xxxx\r\n"
    +"邮箱: 1147417467\r\n"
    +"职位: 产品经理、高级研发工程师\r\n"
    +"<a href='http://blog.csdn.net/myfmyfmyfmyf'>博客点击查看详情</a>";
WxTextMessage txtMessage=new WxTextMessage("muyunfei", null, null, "text",
"0", context, "0");
WxUtil util=new WxUtil();
System.out.println(util.sendReqMsg((ReqBaseMsg) txtMessage));

```

执行效果图如图 3.3 所示。

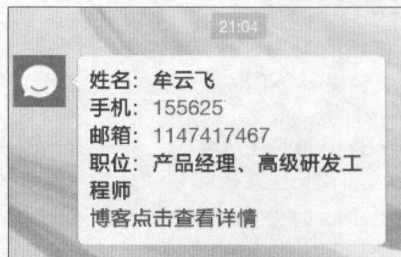


图 3.3 发送 Text 消息



使用技巧:Text 消息支持内容换行(\r\n)和超链接(,一定要加 http://),由于文本文字有字数限制,因此建议使用超链接查看详细内容。用服务号开发过模板消息的读者请注意,模板消息换行是\\r\\n,例如,“\\r\\n 感谢您的光临!\\r\\n 若您交易异常,请拨打 123456 转人工\\r\\n★最新优惠★优惠福利,充 100 减 10 元”。

3.5.3 推送图片消息

图片类型消息（也称 Image 消息），允许消息保密，详细说明如下。

(1) 消息请求结构体:

```
{
  "touser": "UserID1|UserID2|UserID3",
  "toparty": " PartyID1 | PartyID2 ",
  "totag": " TagID1 | TagID2 ",
  "msgtype": "image",
  "agentid": 1,
  "image": {
    "media_id": "MEDIA_ID"
  },
  "safe": "0"
}
```

(2) 参数说明。详细说明如表 3.7 所示。

表 3.7 Image消息体参数说明

参数	是否必需	说明
touser	否	成员ID列表（消息接收者，多个接收者用‘ ’分隔，最多支持1000个）。特殊情况：指定为@all时，则向关注该企业应用的全部成员发送
toparty	否	部门ID列表，多个接收者用‘ ’分隔，最多支持100个。当touser为@all时忽略本参数
totag	否	标签ID列表，多个接收者用‘ ’分隔。当touser为@all时忽略本参数
msgtype	是	消息类型，此时固定为：image。不支持主页型应用
agentid	是	企业应用的ID，整型。可在应用的设置页面查看
media_id	是	图片媒体文件ID，可以调用上传临时素材或者永久素材接口获取，永久素材media_id必须由发消息的应用创建
safe	否	表示是否是保密消息：0表示否，1表示是，默认是0

(3) 示例代码。

此类封装是最普通的封装方法，在方法内以直接拼接字符串的形式完成消息的推送（建议与 Text 消息封装一样封装实体）：

```
/**
 * 发送 Image 消息，发送成功返回 true，失败返回 false
```

```

* @param token 票据
* @param mediaId 多媒体 ID
* @return
*/
public static boolean sendImageMsg(String token,String mediaId){
    boolean flag=false;
    try {
        //测试 mediaId-----
        //首先将图片上传获得 media_id, 有效时间为 5 天
        JSONObject json= send(token,"image","E:/404.png");
        System.out.println(json.toString());
        mediaId=json.getString("media_id");
        //-----
        String imageJson="{\"touser\": \"muyunfei\", \"
            +\"msgtype\": \"image\", \"
            +\"agentid\": \"0\", \"
            +\"image\": {\"
            +\"    \"media_id\": \"\"+mediaId+\"\"
            +\"}, \"
            +\"safe\": \"0\"\"
            +\"}\"";
        sendTextMsg(token,imageJson);
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return flag;
}

```

执行效果图如图 3.4 所示。



图 3.4 发送 Image 消息

备注：多媒体文件的上传将在 3.6 节中介绍。

3.5.4 推送语音消息

语音类型消息（也称 Voice 消息），允许消息保密，消息详细说明如下。

(1) 消息请求结构体:

```
{
    "touser": "UserID1|UserID2|UserID3",
    "toparty": " PartyID1 | PartyID2 ",
    "totag": " TagID1 | TagID2 ",
    "msgtype": "voice",
    "agentid": 1,
    "voice": {
        "media_id": "MEDIA_ID"
    },
    "safe": "0"
}
```

(2) 参数说明。详细说明如表 3.8 所示。

表 3.8 Voice消息体参数说明

参数	是否必需	说明
touser	否	成员ID列表（消息接收者，多个接收者用‘ ’分隔，最多支持1000个）。特殊情况：指定为@all时，则向关注该企业应用的全部成员发送
toparty	否	部门ID列表，多个接收者用‘ ’分隔，最多支持100个。当touser为@all时忽略本参数
totag	否	标签ID列表，多个接收者用‘ ’分隔。当touser为@all时忽略本参数
msgtype	是	消息类型，此时固定为：voice。不支持主页型应用
agentid	是	企业应用的ID，整型。可在应用的设置页面查看
media id	是	语音文件ID，可以调用上传临时素材或者永久素材接口获取
safe	否	表示是否是保密消息：0表示否，1表示是，默认是0

(3) 封装方法示例代码:

```
package myf.caption3.demo3_5;
/**
 * 语音消息
 * @author muyunfei
 */
public class WxVoiceMessage {
    //touser 为微信中添加的账号，并非个人微信号
    //成员 ID 列表（消息接收者，多个接收者用'|'分隔，最多支持 1000 个）
    //特殊情况：指定为@all 时，则向关注该企业应用的全部成员发送
    private String touser;
    //部门 ID 列表，多个接收者用'|'分隔，最多支持 100 个。当 touser 为@all 时忽略本参数
    private String toparty;
    // 标签 ID 列表，多个接收者用'|'分隔。当 touser 为@all 时忽略本参数
    private String totag;
    //消息类型，文本类型为：text、image、voice、video、file、news、mapnews
```

```
private String msgtype;
//企业号应用 ID
private String agentid;
//消息内容, 自行封装类 Media, 只有唯一属性 String media_id
private Media voice;
//表示是否是保密消息: 0 表示否, 1 表示是, 默认是 0
private String safe;

public String getTouser() {
    return touser;
}
public void setTouser(String touser) {
    this.touser = touser;
}
public String getToparty() {
    return toparty;
}
public void setToparty(String toparty) {
    this.toparty = toparty;
}
public String getTotag() {
    return totag;
}
public void setTotag(String totag) {
    this.totag = totag;
}
public String getMsgtype() {
    return msgtype;
}
public void setMsgtype(String msgtype) {
    this.msgtype = msgtype;
}
public String getAgentid() {
    return agentid;
}
public void setAgentid(String agentid) {
    this.agentid = agentid;
}
public String getSafe() {
    return safe;
}
public void setSafe(String safe) {
    this.safe = safe;
}
public Media getVoice() {
    return voice;
}
```



```

    }
    public void setVoice(Media voice) {
        this.voice = voice;
    }

    //有参构造下必须加无参数构造,如果有参构造加 super() 则可以不用写
    public WxVoiceMessage(){}
    //有参构造
    public WxVoiceMessage(String touser, String toparty, String totag,
        String msgtype, String agentid, String mediaId, String safe) {
        this.touser = touser;
        this.toparty = toparty;
        this.totag = totag;
        this.msgtype = msgtype;
        this.agentid = agentid;
        this.voice = new Media(mediaId);
        this.safe = safe;
    }
}

```

本节是将 Voice 消息封装成一个独立的实体类,不继承类不实现任何接口,通过 JSON 包中的 `JSONObject.fromObject(Object obj)` 方法,将实体类转换成 JSON 格式字符串,然后发送信息,示例代码如下:

```

//发送音频消息
String mediaIdString="输入语音素材 Id ";
WxVoiceMessage voiceMessage = new WxVoiceMessage("muyunfei", null, null,
"voice", "0", mediaIdString, "0");
String jsonString=JSONObject.fromObject(voiceMessage).toString();
System.out.println(jsonString);
wxUtil.sendReqMsg(jsonString);

```

备注: 这里的 `public JSONObject sendReqMsg(String jsonContext)` 与 3.5.1 节介绍的方法类似,只需修改方法参数稍作调整即可。如果需要将 JSON 格式的字符串转变成实体类,则可以使用 `ObjectMapper()` 类中的 `readValue` 方法实现。

```
WxVoiceMessage msg=new ObjectMapper().readValue(jsonStr, WxVoiceMessage.class);
```

效果图如图 3.5 所示。

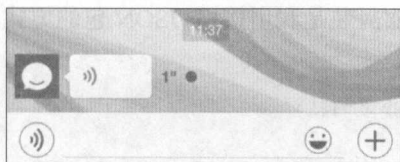


图 3.5 发送 Voice 消息

3.5.5 推送视频消息

视频类型消息（也称 Video 消息），允许消息保密，消息详细说明如下。

(1) 消息请求结构体:

```
{
  "touser": "UserID1|UserID2|UserID3",
  "toparty": " PartyID1 | PartyID2 ",
  "totag": " TagID1 | TagID2 ",
  "msgtype": "video",
  "agentid": 1,
  "video": {
    "media_id": "MEDIA_ID",
    "title": "Title",
    "description": "Description"
  },
  "safe": "0"
}
```

(2) 参数说明。详细说明如表 3.9 所示。

表 3.9 Video消息体参数说明

参数	是否必需	说明
touser	否	成员ID列表（消息接收者，多个接收者用‘ ’分隔，最多支持1000个）。特殊情况：指定为@all时，则向关注该企业应用的全部成员发送
toparty	否	部门ID列表，多个接收者用‘ ’分隔，最多支持100个。当touser为@all时忽略本参数
totag	否	标签ID列表，多个接收者用‘ ’分隔。当touser为@all时忽略本参数
msgtype	是	消息类型，此时固定为：video。不支持主页型应用
agentid	是	企业应用的ID，整型。可在应用的设置页面查看
media_id	是	视频媒体文件ID，可以调用上传临时素材或者永久素材接口获取。不超过20MB，格式：rm、rmvb、wmv、avi、mpg、mpeg、mp4
title	否	视频消息的标题，不得超过128字节，超过会自动截断
description	否	视频消息的描述，不得超过512字节，超过会自动截断
safe	否	表示是否是保密消息：0表示否，1表示是，默认是0

(3) 示例代码。

接口类 IReqMsg.java，本次封装是利用接口+JSON 包的方式实现的，示例代码如下：

```
/**
 * 所有主动调用模式下信息推送的接口
 * @author muyunfei
 */
public interface IReqMsg {
```

```
}
```

接口实现类 WxVideoMessageImpl.java, 实现接口 IReqMsg:

```
package myf.caption3.demo3_5;

/**
 * 语音消息
 * @author muyunfei
 */
public class WxVideoMessageImpl implements IReqMsg {
    //touser 为微信中添加的账号, 并非个人微信号
    //成员 ID 列表 (消息接收者, 多个接收者用'|'分隔, 最多支持 1000 个)。
    //特殊情况: 指定为@all 时, 则向关注该企业应用的全部成员发送
    private String touser;
    //部门 ID 列表, 多个接收者用'|'分隔, 最多支持 100 个。当 touser 为@all 时忽略本参数
    private String toparty;
    //标签 ID 列表, 多个接收者用'|'分隔。当 touser 为@all 时忽略本参数
    private String totag;
    //消息类型, 文本类型为: text、image、voice、video、file、news、mapnews
    private String msgtype;
    //企业号应用 ID
    private String agentid;
    //消息内容
    private VideoMedia video;
    //表示是否是保密消息: 0 表示否, 1 表示是, 默认是 0
    private String safe;

    public String getTouser() {
        return touser;
    }

    public void setTouser(String touser) {
        this.touser = touser;
    }

    public String getToparty() {
        return toparty;
    }

    public void setToparty(String toparty) {
        this.toparty = toparty;
    }

    public String getTotag() {
        return totag;
    }

    public void setTotag(String totag) {
        this.totag = totag;
    }

    public String getMsgtype() {
```

```
        return msgtype;
    }
    public void setMsgtype(String msgtype) {
        this.msgtype = msgtype;
    }
    public String getAgentid() {
        return agentid;
    }
    public void setAgentid(String agentid) {
        this.agentid = agentid;
    }
    public String getSafe() {
        return safe;
    }
    public void setSafe(String safe) {
        this.safe = safe;
    }
    public VideoMedia getVideo() {
        return video;
    }
    public void setVideo(VideoMedia video) {
        this.video = video;
    }
}

//有参构造下必须加无参数构造
public WxVideoMessageImpl(){}
//有参构造
public WxVideoMessageImpl(String touser, String toparty, String totag,
    String msgtype, String agentid, String mediaId, String title, String
description, String safe) {
    this.touser = touser;
    this.toparty = toparty;
    this.totag = totag;
    this.msgtype = msgtype;
    this.agentid = agentid;
    this.video = new VideoMedia(mediaId, title, description);
    this.safe = safe;
}
}
```

媒体类 VideoMedia.java:

```
package myf.caption3.demo3_5;
/**
 * 多媒体 Video 文件类
 * @author muyunfei
 */
```



```

public class VideoMedia {
    //语音文件 ID, 可以调用上传临时素材或者永久素材接口获取
    private String media_id;
    //视频消息的标题, 不得超过 128 字节, 超过会自动截断。非必需
    private String title;
    //视频消息的描述, 不得超过 512 字节, 超过会自动截断。非必需
    private String description;
    public VideoMedia(String mediaId, String title, String description) {
        super();
        this.media_id = mediaId;
        this.title = title;
        this.description = description;
    }
    public String getMedia_id() {
        return media_id;
    }
    public void setMedia_id(String mediaId) {
        media_id = mediaId;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }
}

```

同样通过接口 (interface) + JSON 的方式也可以节约开发时间, 并减少字符串拼接造成的失误, 运行代码如下:

```

//发送视频消息
String
mediaIdString="2RefB1D0jOPDPJpjpXRYgGA6HHVdrM88FVQhalvhceyFViUUma79r_ESqkRzj
SAXxjbUIAqePKqrHvKsI_WaFzw";
IReqMsg voiceMessage = new WxVideoMessageImpl("muyunfei", null, null,
"video", "0", mediaIdString, "测试标题", "测试内容……", "0");
wxUtil.sendReqMsg(voiceMessage);

```

运行效果如图 3.6 所示。



图 3.6 发送 Video 消息



使用技巧：接口类首字母 I（大写），实现类不加 I，后尾名增加 impl。将 new 出的实现类赋予接口或父类，原方法执行不会出现差错，并会增强函数的利用率。

3.5.6 推送文件消息

文件类型消息（也称 File 消息），允许消息保密，消息详细说明如下。

(1) 消息请求结构体：

```
{
  "touser": "UserID1|UserID2|UserID3",
  "toparty": " PartyID1 | PartyID2 ",
  "totag": " TagID1 | TagID2 ",
  "msgtype": "file",
  "agentid": 1,
  "file": {
    "media_id": "MEDIA_ID"
  },
  "safe": "0"
}
```

(2) 参数说明。详细说明如表 3.10 所示。

表 3.10 File消息体参数说明

参数	是否必需	说明
touser	否	成员ID列表（消息接收者，多个接收者用 ‘ ’ 分隔，最多支持 1000个）。特殊情况：指定为@all时，则向关注该企业应用的全部成员发送
toparty	否	部门ID列表，多个接收者用 ‘ ’ 分隔，最多支持100个。当touser为@all时忽略本参数

续表

参数	是否必需	说明
totag	否	标签ID列表，多个接收者用‘ ’分隔。当touser为@all时忽略本参数
msgtype	是	消息类型，此时固定为：file。不支持主页型应用
agentid	是	企业应用的ID，整型。可在应用的设置页面查看
media_id	是	媒体文件ID，可以调用上传临时素材或者永久素材接口获取
safe	否	表示是否是保密消息：0表示否，1表示是，默认是0

(3) 示例代码。

File 消息的消息体结构与 Voice 消息体结构极其相似，这里直接采用 Voice 消息的处理方式，通过 JSON 转变实体为 JSON 字符串，实体类代码如下：

```
package myf.caption3.demo3_5;
/**
 * 语音消息
 * @author muyunfei
 */
public class WxFileMessage {

    //touser 为微信中添加的账号，并非个人微信号
    //成员 ID 列表（消息接收者，多个接收者用'|'分隔，最多支持 1000 个）
    //特殊情况：指定为@all 时，则向关注该企业应用的全部成员发送
    private String touser;
    //部门 ID 列表，多个接收者用'|'分隔，最多支持 100 个。当 touser 为@all 时忽略本参数
    private String toparty;
    //标签 ID 列表，多个接收者用'|'分隔。当 touser 为@all 时忽略本参数
    private String totag;
    //消息类型，文本类型为：text、image、voice、video、file、news、mapnews
    private String msgtype;
    //企业号应用 ID
    private String agentid;
    //消息内容
    private Media file;
    //表示是否是保密消息：0 表示否，1 表示是，默认是 0
    private String safe;

    public String getTouser() {
        return touser;
    }
    public void setTouser(String touser) {
        this.touser = touser;
    }
    public String getToparty() {
        return toparty;
    }
```

```
    }  
    public void setToparty(String toparty) {  
        this.toparty = toparty;  
    }  
    public String getTotag() {  
        return totag;  
    }  
    public void setTotag(String totag) {  
        this.totag = totag;  
    }  
    public String getMsgtype() {  
        return msgtype;  
    }  
    public void setMsgtype(String msgtype) {  
        this.msgtype = msgtype;  
    }  
    public String getAgentid() {  
        return agentid;  
    }  
    public void setAgentid(String agentid) {  
        this.agentid = agentid;  
    }  
    public String getSafe() {  
        return safe;  
    }  
    public void setSafe(String safe) {  
        this.safe = safe;  
    }  
    public Media getFile() {  
        return file;  
    }  
    public void setFile(Media file) {  
        this.file = file;  
    }  
}  
  
//有参构造下必须加无参数构造  
public WxFileMessage() {}  
//有参构造  
public WxFileMessage(String touser, String toparty, String totag,  
    String msgtype, String agentid, String mediaId, String safe) {  
    this.touser = touser;  
    this.toparty = toparty;  
    this.totag = totag;  
    this.msgtype = msgtype;  
    this.agentid = agentid;  
    this.file = new Media(mediaId);  
}
```

```

        this.safe = safe;
    }
}

```

执行 File 消息推送，代码如下：

```

//发送文件消息
String
mediaIdString="20DdvT1n1JQfps48keCbEWkY4cOw9ivL5VnK-9xLpn5yYsFVzOv-h3wtEem2M
Pbggvq26HR1eFbDeDeWGbyLPHA";
WxFileMessage voiceMessage = new WxFileMessage("muyunfei", null, null,
"file", "0", mediaIdString, "0");
String jsonString=JSONObject.fromObject(voiceMessage).toString();
System.out.println(jsonString);
wxUtil.sendReqMsg(jsonString);

```

效果图如图 3.7 所示。

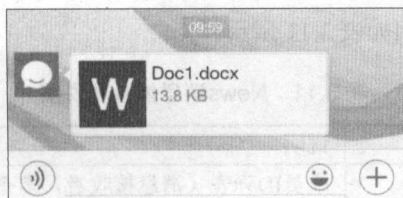


图 3.7 发送 File 消息

注意：使用 JSON 包中的 `JSONObject.fromObject(Object object)` 方法，实体类属性必须有 GET 方法，否则会出现异常，实体属性建议 `get`、`set` 都加。

3.5.7 推送新闻消息

新闻消息（也称 News 消息，图文消息），不支持消息保密，消息内容不在微信后台存储，但 News 消息在企业号开发过程中却使用得比较频繁，对于数据安全方面则需要读者自行开发相应的安全策略（本书第 8 章会介绍相应的数据安全）。News 消息详细说明如下。

注意：News 消息是主动模式消息推送中唯一一个不支持保密的消息类型。

（1）消息请求结构体：

```

{
    "touser": "UserID1|UserID2|UserID3",
    "toparty": " PartyID1 | PartyID2 ",
    "totag": " TagID1 | TagID2 ",
    "msgtype": "news",
    "agentid": 1,
    "news": {

```

```

    "articles": [
      {
        "title": "Title",
        "description": "Description",
        "url": "URL",
        "picurl": "PIC_URL"
      },
      {
        "title": "Title",
        "description": "Description",
        "url": "URL",
        "picurl": "PIC_URL"
      }
    ]
  }
}
```

(2) 参数说明。详细说明如表 3.11 所示。

表 3.11 News消息体参数说明

参数	是否必需	说明
touser	否	成员ID列表（消息接收者，多个接收者用‘ ’分隔，最多支持1000个）。特殊情况：指定为@all时，则向关注该企业应用的全部成员发送
toparty	否	部门ID列表，多个接收者用‘ ’分隔，最多支持100个。当touser为@all时忽略本参数
totag	否	标签ID列表，多个接收者用‘ ’分隔。当touser为@all时忽略本参数
msgtype	是	消息类型，此时固定为：news。不支持主页型应用
agentid	是	企业应用的ID，整型。可在应用的设置页面查看
articles	是	图文消息，一个图文消息支持1到10条图文
title	否	标题，不得超过128字节，超过会自动截断
description	否	描述，不得超过512字节，超过会自动截断
url	否	点击后跳转的链接
picurl	否	图文消息的图片链接，支持JPG、PNG格式，较好的效果为大图640×320，小图80×80。如不填，则在客户端不显示图片

! 注意：articles 中 News 消息不能超过 10 条（官网写了 8 条，实际上是 10 条），超过之后将显示 45008 超出文章数限制的错误{"errcode":45008,"errmsg":"article size out of limit"}；picurl 图片链接可以是图片地址链接，也可以是图片输出流的链接。

(3) 示例代码。

News 消息实体类 WxNewsMessage.java 代码如下：

```
package myf.caption3.demo3_5;
```



```

import java.util.List;
public class WxNewsMessage extends ReqBaseMsg {
    public WxNewsMessage() {}
    //消息
    public List<WxArticle> news;
    public List<WxArticle> getNews() {
        return news;
    }
    public void setNews(List<WxArticle> news) {
        this.news = news;
    }
    public WxNewsMessage(String touser, String toparty, String totag,
        String msgtype, String agentid) {
        super();
        super.touser = touser;
        super.toparty = toparty;
        super.totag = totag;
        super.msgtype = msgtype;
        super.agentid = agentid;
    }

    public String toJsonStr(){
        StringBuffer jsonStr=new StringBuffer("{}");
        StringBuffer str_tmp= new StringBuffer("");
        //{"touser": "@all","msgtype": "text","agentid": "0","\
"text": {"content": "helloworld"},"safe": "0"};
        if(null!=touser&&!"".equals(touser)){
            if(!"".equals(str_tmp.toString())){
                str_tmp.append(",");
            }
            str_tmp.append("\"touser\": \""+touser+"\"");
        }
        if(null!=toparty&&!"".equals(toparty)){
            if(!"".equals(str_tmp.toString())){
                str_tmp.append(",");
            }
            str_tmp.append("\"toparty\": \""+toparty+"\"");
        }
        if(null!=totag&&!"".equals(totag)){
            if(!"".equals(str_tmp.toString())){
                str_tmp.append(",");
            }
            str_tmp.append("\"totag\": \""+totag+"\"");
        }
        if(null!=msgtype&&!"".equals(msgtype)){
            //去除空格

```



```

msgtype=msgtype.trim();
//判断是否加逗号
if(!"".equals(str_tmp.toString())){
    str_tmp.append(",");
}
str_tmp.append("\"msgtype\": \""+msgtype+"\"");
//新闻消息
if(null!=news&&0!=news.size()){
    if(!"".equals(str_tmp.toString())){
        str_tmp.append(",");
    }
    StringBuffer content=new StringBuffer("");
    for (int i = 0; i < news.size(); i++) {
        if(i!=0){
            content.append(",");
        }
        content.append("{");
        //获得一条消息
        WxArticle info = news.get(i);
        StringBuffer articleContent=new StringBuffer("");
        if(null!=info.getTitle()&&!"".equals(info.getTitle())){
            if(null!=articleContent.toString()&&!"".equals
(articleContent.toString())){
                articleContent.append(",");
            }
            //标题
            articleContent.append("\"title\": \""+info.getTitle()+
"\"");
        }
        if(null!=info.getDescription()&&!"".equals
(info.getDescription())){
            if(null!=articleContent&&!"".equals(articleContent)){
                articleContent.append(",");
            }
            //描述
            articleContent.append("\"description\":
\""+info.getDescription()+"\"");
        }
        if(null!=info.getUrl()&&!"".equals(info.getUrl())){
            if(null!=articleContent&&!"".equals(articleContent)){
                articleContent.append(",");
            }
            //详细地址
            articleContent.append("\"url\":
\""+info.getUrl()+"\"");
        }
    }
}

```

```

        if (null != info.getPicurl() && !"".equals(
(info.getPicurl())) {
            if (null != articleContent && !"".equals(articleContent)) {
                articleContent.append(",");
            }
            // 图片地址
            articleContent.append("\"picurl\": 
\"" + info.getPicurl() + "\"");
        }
        content.append(articleContent.toString() + "}");
    }
    str_tmp.append("\"news\": {\"articles\": [" + content + "]}");
}

}

if (null != agentid && !"".equals(agentid)) {
    if (!"".equals(str_tmp.toString())) {
        str_tmp.append(",");
    }
    str_tmp.append("\"agentid\": \"" + agentid + "\"");
}

}

jsonStr.append(str_tmp);
jsonStr.append(")");
return jsonStr.toString();
}
}

```

文章类 WxArticle.java, 代码如下:

```

package myf.caption3.demo3_5;
/**
 * 文章类
 */
public class WxArticle {
    // 图文消息名称
    private String title;
    // 图文消息描述
    private String description;
    // 图片链接, 支持 JPG、PNG 格式, 较好的效果为大图 640×320, 小图 80×80
    private String picurl;
    // 点击图文消息跳转链接
    private String url;
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {

```

```
        this.title = title;
    }
    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }
    public String getPicurl() {
        return picurl;
    }
    public void setPicurl(String picurl) {
        this.picurl = picurl;
    }
    public String getUrl() {
        return url;
    }
    public void setUrl(String url) {
        this.url = url;
    }
    public WxArticle(String title, String description, String picurl, String
url) {
        super();
        this.title = title;
        this.description = description;
        this.picurl = picurl;
        this.url = url;
    }
}
```

News 消息推送，代码执行如下：

```
//发送新闻消息
//创建 News 消息
WxNewsMessage news = new WxNewsMessage("muyunfei", null, null, "news", "0");
WxArticle article1=new WxArticle("CSDN 博客",
    "测试发送新闻信息。。。。" +
    "旨在提供企业移动应用入口,它可以帮助企业建立员工",    "http://myfmyfmyfmyf.
vicp.cc /newsMobileAction.do?action=getImage&imageId=1",
    "http://blog.csdn.net/myfmyfmyfmyf");

WxArticle article2=new WxArticle("微信企业号",
    "微信企业号是微信为企业客户提供的移动服务, " +
    "旨在提供企业移动应用入口, 它可以帮助企业建立员工",
    "http://avatar.csdn.net/4/1/B/1_myfmyfmyfmyf.jpg",
    "http://blog.csdn.net/myfmyfmyfmyf");
```

```

WxArticle article3=new WxArticle("测试信息",
    "微信企业号是微信为企业客户提供的移动服务," +
    "旨在提供企业移动应用入口。它可以帮助企业建立员工",
    " http:// myfmyfmyf.vicp.cc /newsMobileAction.do?action=
getImage&imageId=2);
List<WxArticle> list = new ArrayList<WxArticle>();
list.add(article1);
list.add(article2);
list.add(article3);
news.setNews(list);
//推送 News 消息
wxUtil.sendReqMsg(news);

```

消息体完成封装之后,在后期的调用中将极大地缩短开发时间,执行效果如图 3.8 所示。本节是通过字符串的拼接方式实现的,可以看出,toJsonStr()方法过于繁琐,容易在开发中出现失误,所以建议使用 JSON 包提供的 JSONObject.fromObject 方法生成企业号所需要的 JSON 格式字符串。



图 3.8 发送 News 消息

使用技巧: News 消息中的封面图片 (picurl), 可以是一张图片的文件链接, 也可以是图片的输出流链接。News 消息的第一条消息, 封面图片将被放大。

3.5.8 推送永久图文消息

MpNews 消息与 News 消息系统都是图文消息, 不同之处在于, MpNews 消息支持消息保密选项, 并且消息内容存储在微信后台。MpNews 消息在数据请求上也与其他消息不同, 分为两种结构: 直接发送消息内容和发送永久图文素材, 详细说明如下。

注意: 每个应用每天最多可以发送 100 次 MpNews 消息。

(1) 直接发送消息内容。

消息请求结构体:

```
{
  "touser": "UserID1|UserID2|UserID3",
  "toparty": " PartyID1 | PartyID2 ",
  "totag": " TagID1 | TagID2 ",
  "msgtype": "mpnews",
  "agentid": 1,
  "mpnews": {
    "articles":[
      {
        "title": "Title",
        "thumb_media_id": "id",
        "author": "Author",
        "content_source_url": "URL",
        "content": "Content",
        "digest": "Digest description",
        "show_cover_pic": "0"
      },
      {
        "title": "Title",
        "thumb_media_id": "id",
        "author": "Author",
        "content_source_url": "URL",
        "content": "Content",
        "digest": "Digest description",
        "show_cover_pic": "0"
      }
    ]
  },
  "safe":"0"
}
```

(2) 参数说明。详细说明如表 3.12 所示。

表 3.12 MpNews直接发送消息内容参数说明

参数	是否必需	说明
touser	否	成员ID列表（消息接收者，多个接收者用‘ ’分隔，最多支持1000个）。特殊情况：指定为@all时，则向关注该企业应用的全部成员发送
toparty	否	部门ID列表，多个接收者用‘ ’分隔，最多支持100个。当touser为@all时忽略本参数
totag	否	标签ID列表，多个接收者用‘ ’分隔。当touser为@all时忽略本参数
msgtype	是	消息类型，此时固定为：mpnews。不支持主页型应用
agentid	是	企业应用的ID，整型。可在应用的设置页面查看

续表

参数	是否必需	说明
articles	是	图文消息，一个图文消息支持1~10条图文
title	否	标题，不得超过128字节，超过会自动截断
thumb_media_id	是	图文消息缩略图的media_id，可以在上传多媒体文件接口中获得。此处thumb_media_id，即为上传接口返回的media_id
author	否	图文消息的作者，不超过64字节
content_source_url	否	图文消息点击“阅读原文”之后的页面链接
content	是	图文消息的内容，支持html标签，不得超过666 KB
digest	否	图文消息的描述，不得超过512字节，超过会自动截断
show_cover_pic	否	是否显示封面：1为显示，0为不显示
safe	否	表示是否是保密消息：0表示否，1表示是，默认是0

(3) 发送永久图文素材。

消息请求结构体：

```
{
  "touser": "UserI1|UserID2|UserID3",
  "toparty": " PartyID1 | PartyID2 ",
  "msgtype": "mpnews",
  "agentid": 1,
  "mpnews": {
    "media_id": "MEDIA_ID"
  },
  "safe": "0"
}
```

(4) 参数说明。详细说明如表 3.13 所示。

表 3.13 MpNews发送永久图文素材参数说明

参数	是否必需	说明
touser	否	成员ID列表（消息接收者，多个接收者用‘ ’分隔，最多支持1000个）。特殊情况：指定为@all时，则向关注该企业应用的全部成员发送
toparty	否	部门ID列表，多个接收者用‘ ’分隔，最多支持100个。当touser为@all时忽略本参数
totag	否	标签ID列表，多个接收者用‘ ’分隔。当touser为@all时忽略本参数
msgtype	是	消息类型，此时固定为：mpnews。不支持主页型应用
agentid	是	企业应用的ID，整型。可在应用的设置页面查看
media_id	是	素材资源标识ID，通过上传永久图文素材接口获得
safe	否	表示是否是保密消息：0表示否，1表示是，默认是0


 **注意：**media_id 必须是在当前应用下创建的永久素材，永久素材的创建必须指定应用 id（素材管理将在 3.5.9 节讲解）。

图 3-1-5 (5) 示例代码。

MpNews 消息与 News 消息基本一致, 这里将采用 JSON 处理的方式推送消息, 以方便读者在代码量上对比字符串拼接与 JSON 包两种方式的差距, MpNews 消息实体类如下。

MpNews 消息主体类 WxMpNewsMessage.java:

```
package myf.caption3.demo3_5;

import java.util.List;

/**
 * @author muyunfei
 * MpNew 消息主体类
 */
public class WxMpNewsMessage {
    //touser 为微信中添加的账号, 并非个人微信号
    //成员 ID 列表 (消息接收者, 多个接收者用'|'分隔, 最多支持 1000 个)
    //特殊情况: 指定为@all 时, 则向关注该企业应用的全部成员发送
    private String touser;
    //部门 ID 列表, 多个接收者用'|'分隔, 最多支持 100 个。当 touser 为@all 时忽略本参数
    private String toparty;
    //标签 ID 列表, 多个接收者用'|'分隔。当 touser 为@all 时忽略本参数
    private String totag;
    //消息类型, 文本类型为 mapnews
    private String msgtype;
    //企业号应用 ID
    private String agentid;
    //消息
    private WxArticleList mpnews;
    //是否保密
    private String safe;

    public WxMpNewsMessage() {}
    public WxMpNewsMessage(String touser, String toparty, String totag,
        String msgtype, String agentid, List<WxMpArticle> mpnews, String safe) {
        this.touser = touser;
        this.toparty = toparty;
        this.totag = totag;
        this.msgtype = msgtype;
        this.agentid = agentid;
        this.mpnews = new WxArticleList(mpnews);
        this.safe = safe;
    }
    public String getTouser() {
        return touser;
    }
    public String getToparty() {
```

```

        return toparty;
    }
    public String getTotag() {
        return totag;
    }
    public String getMsgtype() {
        return msgtype;
    }
    public String getAgentid() {
        return agentid;
    }
    public String getSafe() {
        return safe;
    }
    public WxArticleList getMpnews() {
        return mpnews;
    }
}

```

MpNews 消息附属类新闻集合类 WxArticleList.java:

```

package myf.caption3.demo3_5;
import java.util.List;
/**
 * @author muyunfei
 * 文章数组类，用于生成 JSON 格式
 */
public class WxArticleList {
    private List<WxMpArticle> articles;
    public WxArticleList(List<WxMpArticle> articles) {
        super();
        this.articles = articles;
    }
    public List<WxMpArticle> getArticles() {
        return articles;
    }
    public void setArticles(List<WxMpArticle> articles) {
        this.articles = articles;
    }
}

```

新闻内容类 WxMpArticle.java:

```

package myf.caption3.demo3_5;

/**
 * @author muyunfei
 * 新闻内容

```

```
*/
public class WxMpArticle {
    //图文消息名称
    private String title;
    //图文消息缩略图的 media_id
    private String thumb_media_id;
    //作者
    private String author;
    //图文消息, 点击“阅读原文”之后的页面链接
    private String content_source_url;
    //内容, 支持html 标签, 不得超过 666 KB
    private String content;
    //图文消息的描述, 不得超过 512 字节, 超过会自动截断
    private String digest;
    //是否显示封面: 1 为显示, 0 为不显示
    private String show_cover_pic;

    public WxMpArticle(String title, String thumbMediaId, String author,
        String contentSourceUrl, String content, String digest,
        String showCoverPic) {
        super();
        this.title = title;
        thumb_media_id = thumbMediaId;
        this.author = author;
        content_source_url = contentSourceUrl;
        this.content = content;
        this.digest = digest;
        show_cover_pic = showCoverPic;
    }

    public String getTitle() {
        return title;
    }

    public String getThumb_media_id() {
        return thumb_media_id;
    }

    public String getAuthor() {
        return author;
    }

    public String getContent_source_url() {
        return content_source_url;
    }

    public String getContent() {
        return content;
    }

    public String getDigest() {
        return digest;
    }
}
```



```

    }
    public String getShow_cover_pic() {
        return show_cover_pic;
    }
}

```

通过 JSON 包的方式，可以减少人为的失误，执行代码如下：

```

//发送 MP 新闻消息
//创建 MpNews 消息
WxMpArticle article1=new WxMpArticle("CSDN 博客","1wSwU1MJFZI-6YasolX-_sB4yNKVM5doGcI8cEQBy6L4yQvYBbOHga_NUTiJQxsIiIYydEQPlMNE_07-XD3wz_Q","牟云飞",
    "http://blog.csdn.net/myfmyfmyfmyf",
    "测试发送新闻信息。。<br/><font color='red'>支持html 标签</font>。。。<br/><a href='http://blog.csdn.net/myfmyfmyfmyf'>欢迎查看博客 http://blog.csdn.net/myfmyfmyfmyf</a>",
    "描述信息","1");

List<WxMpArticle> list = new ArrayList<WxMpArticle>();
list.add(article1);
WxMpNewsMessage news = new WxMpNewsMessage("muyunfei", null, null, "mpnews",
"0",list,"1");
//推送 MpNews 消息
System.out.println(JSONObject.fromObject(news).toString());
wxUtil.sendReqMsg(JSONObject.fromObject(news).toString());

```

效果图如图 3.9 所示。



图 3.9 发送 News 消息



注意：thumb_media_id 为 mediaId，不可以写成 URL 链接。JSON 包的使用、类属性必须增加 GET 方法。JSON 格式中，{} 表示对象，[] 表示数组或者集合，发送的消息内容支持 HTML 标签；保密的 MpNews 消息会加上水印。

3.5.9 管理端推送消息

微信企业号在提供接口的同时，还提供了管理端。登录管理端，单击“发消息”，就能够直接发送信息，如图 3.10 和图 3.11 所示。可以根据消息选择不同的消息类型进行发送，不过还是建议根据业务需要开发自己的管理端，以方便后台信息管理。对于数据安全以及内外网分离的客户而言，更适合定制化开发管理端。



图 3.10 管理端发送 News 消息



图 3.11 管理端发送视频消息



备注：企业号管理端最新提供了定时发送的功能，不过暂无相应开发接口。

3.6 素材管理

企业号可以使用素材管理接口将多媒体文件包括图片、音频、视频、文件以及图文消息上传到素材库，获得 `media_id`，通过 `media_id` 对多媒体文件、多媒体消息素材进行获取和调用等操作。通过对本节的学习，读者应了解素材管理，掌握素材管理的接口及其应用。

3.6.1 接口说明

微信企业号素材文件主要分为两种：

(1) 临时素材文件

临时素材 (`media_id`) 在上传到微信服务器之后，只能保存 3 天时间。3 天内 `media_id` 一直有效，3 天过后将自动删除 (`media_id`)，以节省企业号服务器资源。

(2) 永久素材文件

永久素材 (`media_id`) 将会一直保存在微信服务器上，但是对企业能够保存的永久素材数量有所限制，整个企业号图文消息素材和图片素材数目的上限为 5000，其他类型为 1000。

本节将详细介绍两种素材文件的上传、获取、管理接口，不过对于大部分客户来说，还是希望数据能够保存在自己的服务器中由于微信企业号的缓存时间及数量都有限制，所以这里建议读者采用数据流的形式显示本地服务文件（将在第 8 章介绍），尽可能地减少上传文件，以达到数据安全以及容灾处理等问题。

注意：主页型应用只能发送 Text 消息，所以主页型应用无法调用上传、获取、删除等操作。

3.6.2 上传临时素材文件

用于上传临时的图片、语音、视频等媒体资源文件以及普通文件（如 doc、ppt），接口调用成功之后，返回媒体资源标识 ID: `media_id`。需要注意的是，临时素材文件的有效期为三天，在有效时间内 `media_id` 是可复用的，同一个 `media_id` 可用于消息的多次发送，接口详细说明如下。

说明：此接口早期的开发者可能熟悉，此接口是早期的“上传多媒体文件”接口。

(1) Https 请求链接如下：

`https://qyapi.weixin.qq.com/cgi-bin/media/upload?access_token=ACCESS_TOKEN&type=TYPE`

(2) 数据请求方式。

POST 方式进行数据请求。

(3) 参数说明。详细说明如表 3.14 所示。

表 3.14 上传临时素材请求参数说明

参数	是否必需	说明
<code>access_token</code>	是	调用接口凭证（需要缓存处理）

续表

参数	是否必需	说明
type	是	媒体文件类型，分别有图片（image）、语音（voice）、视频（video）和普通文件（file）
Post消息体media	是	文件数据流，form-data中媒体文件标识，有filename、filelength、content-type等信息

（4）权限说明。

完全公开。所有管理组均可调用，media_id 可以共享。

（5）返回参数说明（详细说明见表 3.15）：

```
{
  "type": "image",
  "media_id": "1G6nrLmr5EC3MMb_-zK1dDdzmd0p7cNliYu9V5w7o8K0",
  "created_at": "1380000000"
}
```

表 3.15 上传临时素材返回参数说明

参数	说明
type	媒体文件类型，分别有图片（image）、语音（voice）、视频（video）和普通文件（file）
media_id	媒体文件上传后获取的唯一标识
created_at	媒体文件上传时间戳

（6）媒体文件上传限制：

- 所有文件大小（size）必须大于 5 字节。
- 图片（image）：2MB，支持 JPG，PNG 格式。
- 语音（voice）：2MB，播放长度不超过 60s，支持 AMR 格式。
- 视频（video）：10MB，支持 MP4 格式。
- 普通文件（file）：20MB。

（7）示例代码：

```
/**
 * 文件上传到微信服务器
 * @param fileType 文件类型为媒体文件类型，分别有图片（image）、语音（voice）、视频（video）和普通文件（file）
 * @param filePath 文件路径
 * @return JSONObject
 * @throws Exception
 */
public JSONObject send(String fileType, String filePath) throws Exception {
    String result = null;
    File file = new File(filePath);
    if (!file.exists() || !file.isFile()) {
```

```

        throw new IOException("文件不存在");
    }
    String token=getTokenFromWx();
    /**
    * 第一部分
    */
    URL urlObj = new URL("https://qyapi.weixin.qq.com/cgi-bin/media/upload?
access_token="+ token
        + "&type="+fileType+"");

    HttpURLConnection con = (HttpURLConnection) urlObj.openConnection();
    con.setRequestMethod("POST"); //以 POST 方式提交表单, 默认 GET 方式
    con.setDoInput(true);
    con.setDoOutput(true);
    con.setUseCaches(false); //POST 方式不能使用缓存
    //设置请求头信息
    con.setRequestProperty("Connection", "Keep-Alive");
    con.setRequestProperty("Charset", "UTF-8");
    //设置边界
    String BOUNDARY = "-----" + System.currentTimeMillis();
    con.setRequestProperty("Content-Type", "multipart/form-data; boundary=
"+ BOUNDARY);
    //请求正文信息
    //第一部分:
    StringBuilder sb = new StringBuilder();
    sb.append("--"); //必须多两道线
    sb.append(BOUNDARY);
    sb.append("\r\n");
    sb.append("Content-Disposition: form-data;name=\"media\";filename=\""+
file.getName() + "\"\r\n");
    sb.append("Content-Type:application/octet-stream\r\n\r\n");
    byte[] head = sb.toString().getBytes("utf-8");
    //获得输出流
    OutputStream out = new DataOutputStream(con.getOutputStream());
    //输出表头
    out.write(head);
    //文件正文部分
    //把文件已流文件的方式 推入到 URL 中
    DataInputStream in = new DataInputStream(new FileInputStream(file));
    int bytes = 0;
    byte[] bufferOut = new byte[1024];
    while ((bytes = in.read(bufferOut)) != -1) {
        out.write(bufferOut, 0, bytes);
    }
    in.close();
    //结尾部分

```



```

byte[] foot = ("\r\n--" + BOUNDARY + "--\r\n").getBytes("utf-8");// 定义最后数据分隔线
out.write(foot);
out.flush();
out.close();
StringBuffer buffer = new StringBuffer();
BufferedReader reader = null;
try {
    // 定义 BufferedReader 输入流来读取 URL 响应
    reader = new BufferedReader(new InputStreamReader(con.getInputStream()));
    String line = null;
    while ((line = reader.readLine()) != null) {
        //System.out.println(line);
        buffer.append(line);
    }
    if(result==null){
        result = buffer.toString();
    }
} catch (IOException e) {
    System.out.println("发送 POST 请求出现异常! " + e);
    e.printStackTrace();
    throw new IOException("数据读取异常");
} finally {
    if(reader!=null){
        reader.close();
    }
}
JSONObject jsonObj =JSONObject.fromObject(result);
return jsonObj;
}

```

3.6.3 获取临时素材文件

通过有效的 media_id 获取图片、语音、视频等文件，在协议上与普通的 http 文件下载完成相同，通过数据流获取文件即可，接口详细说明如下。

(1) Https 请求链接如下：

https://qyapi.weixin.qq.com/cgi-bin/media/get?access_token=ACCESS_TOKEN&media_id=MEDIA_ID

(2) 数据请求方式。

GET 方式进行数据请求。

(3) 参数说明。详细说明如表 3.16 所示。

表 3.16 获取临时素材请求参数说明


参数	是否必需	说明
access_token	是	调用接口凭证（需要缓存处理）
media_id	是	媒体文件唯一标识

(4) 权限说明。

完全公开。所有管理组均可调用，media_id 可以共享。

(5) 返回参数说明。

和普通的 http 下载相同，请根据 http 头做出相应的处理，如文件类型（Content-Type）、文件名（filename），等等。



使用技巧：通过 con.getContentType()方法获得图片文件类型，以便进行文件保存。

正确返回数据流，头部信息：

```
{
  HTTP/1.1 200 OK
  Connection: close
  Content-Type: image/jpeg
  Content-disposition: attachment; filename="MEDIA_ID.jpg"
  Date: Sun, 06 Jan 2013 10:20:18 GMT
  Cache-Control: no-cache, must-revalidate
  Content-Length: 339721

  Xxxx
}
```

错误返回信息 JSON 字符串：

```
{
  "errcode": "40004",
  "errmsg": "invalid media_id"
}
```

(6) 示例代码：

```
/**
 * 下载临时文件
 * @param mediaId 媒体文件 ID
 * @param resp 数据响应
 * @return InputStream 获得文件输入流，注释部分为直接保存本地
 */
public InputStream downloadFile(String mediaId,HttpServletResponse resp){
    //获取 Token 凭证
    String token=getTokenFromWx();
    String
    urlStr="https://qyapi.weixin.qq.com/cgi-bin/media/get?access_token="+token+"
}
```

```
&media_id="+mediaId;
    try {
        URL urlObj = new URL(urlStr);
        HttpURLConnection con = (HttpURLConnection) urlObj.openConnection();
        con.setDoInput(true);
        Map<String, List<String>> aa = con.getHeaderFields();
        //设置 Servlet 请求文件格式
        String contentType = con.getContentType();
        resp.setContentType(contentType);
        //返回输出流
        return con.getInputStream();
    }
    /**
    //保存文件
    InputStream in = util.downloadFile(accessId, resp);
    if (null != in) {
        OutputStream outputStream = new FileOutputStream(new File("G:\\
app\\aaa.jpg"));
        byte[] bytes = new byte[1024];
        int cnt=0;
        while ((cnt=in.read(bytes,0,bytes.length)) != -1) {
            outputStream.write(bytes, 0, cnt);
        }
        outputStream.flush();
        outputStream.close();
        in.close();
    }else{
        //图片获取失败, 显示默认图片
        System.out.println("图片获取失败");
    }
    **/
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
return null;
}
```

3.6.4 上传永久素材（非图文素材）

素材文件分为永久图文素材和永久非图文素材，非图文素材文件与上传临时文件相似，接口详细说明如下。

(1) Https 请求链接如下：

https://qyapi.weixin.qq.com/cgi-bin/material/add_material?agentid=AGENTID&type=TYPE&access_token=ACCESS_TOKEN

注意：上传永久素材时需用到应用的 ID（agentId），上传临时素材则不需要。

(2) 数据请求方式。

POST 方式进行数据请求。

(3) 参数说明。详细说明如表 3.17 所示。

表 3.17 上传永久素材（非图文）请求参数说明

参数	是否必需	说明
access_token	是	调用接口凭证（需要缓存处理）
type	是	媒体文件类型，分别有图片（image）、语音（voice）、视频（video）和普通文件（file）
Post消息体media	是	文件数据流，form-data中媒体文件标识，有filename、filelength、content-type等信息

(4) 权限说明。

管理组须拥有应用（agent）的使用权限；media_id 只可以在同一个应用的不同管理组共享。

(5) 返回参数说明（详细说明如表 3.18 所示）：

```
{
  "errcode":0,
  "errmsg":"ok",
  "media_id": "2-G6nrLmr5EFSDC3MMfasdfb_-zK1dDdzmd0p7"
}
```

表 3.18 上传永久素材（非图文）返回参数说明

参数	说明
errcode	返回结果，0成功，1失败
errmsg	返回结果说明
media_id	媒体ID

(6) 媒体文件上传限制：

所有文件大小（size）必须大于 5 字节。

图片（image）：2MB，支持 JPG、PNG 格式。

语音（voice）：2MB，播放长度不超过 60s，支持 AMR 格式。

视频（video）：10MB，支持 MP4 格式。

普通文件（file）：20MB。

整个企业图文消息素材和图片素材数目的上限为 5000，其他类型为 1000，超出数量将返回 45028 错误码。示例代码参照“3.6.2 上传临时素材文件”。

3.6.5 上传永久素材（图文素材）

上传永久图文素材，接口将返回 media_id。media_id 用于发送 MpNews 消息，详细接口说明如下。

(1) Https 请求链接如下:

https://qyapi.weixin.qq.com/cgi-bin/material/add_mpnews?access_token=ACCESS_TOKEN



注意: 上传永久图文素材需要应用的 ID (agentId) 在消息体中拼接, 上传其他永久素材则在 URL 链接中拼接应用 ID。

(2) 数据请求方式。

POST 方式进行数据请求。

(3) 消息请求结构体:

```
{
  "agentid":0,
  "mpnews":{
    "articles":[
      {
        "title": "Title01",
        "thumb_media_id": "2-G6nrLmr5EC3MMb_-zK1dDdzmd0p7cNliYu9V5w7o8K0",
        "author": "zs",
        "content_source_url": "",
        "content": "Content001",
        "digest": "airticle01",
        "show_cover_pic": "0"
      },
      {
        "title": "Title02",
        "thumb_media_id": "2-G6nrLmr5EC3MMb_-zK1dDdzmd0p7",
        "author": "Author001",
        "content_source_url": "",
        "content": "Content002",
        "digest": "article02",
        "show_cover_pic": "0"
      }
    ]
  }
}
```

//此处有多篇文章, 最多 10 篇

参数详细说明如表 3.19 所示。

表 3.19 上传永久素材 (图文) 请求参数说明

参数	是否必需	说明
access_token	是	调用接口凭证 (需要缓存处理)
agentid	是	企业应用的ID, 整型, 可在应用的设置页面查看
articles	是	图文消息, 一个图文消息支持1到10个图文
title	是	图文消息的标题
thumb_media_id	是	图文消息缩略图的media_id, 可以在上传永久素材接口中获得
author	否	图文消息的作者

续表

参数	是否必需	说明
content_source_url	否	图文消息点击“阅读原文”之后的页面链接
content	是	图文消息的内容，支持html标签
digest	否	图文消息的描述
show_cover_pic	否	是否显示封面，1为显示，0为不显示。默认为0

(4) 权限说明。

管理组须拥有应用（agent）的使用权限；media_id 只可以在同一个应用的不同管理组共享。

(5) 返回参数说明（详细说明如表 3.20 所示）：

```
{
  "errcode":0,
  "errmsg":"ok",
  "media_id": "2-G6nrLmr5EFSDC3MMfasdfb_-zK1dDdzmd0p7"
}
```

表 3.20 上传永久素材（图文）返回参数说明

参数	说明
errcode	返回结果：0成功，1失败
errmsg	返回结果说明
media_id	图文素材ID

(6) 示例代码：

```
package myf.caption3.demo3_5;
import java.util.List;
/**
 * @author 牟云飞
 * 上传永久图文素材
 */
public class WxMpNewsSource {
    //企业号应用 ID
    private String agentid;
    //消息，参考 3.5.8 MpNews 消息
    private WxArticleList mpnews;

    public WxMpNewsSource(){}
    public WxMpNewsSource( String agentid,List<WxMpArticle> mpnews) {

        this.agentid = agentid;
        this.mpnews=new WxArticleList (mpnews);
    }
    //GET 方法
    public String getAgentid() {
        return agentid;
    }
    public WxArticleList getMpnews() {
```



```
        return mpnews;
    }
}
```

备注：其他实体类 WxArticleList，请参照“3.5.8 MpNews 消息”。

执行代码如下所示：

```
//上传永久图文素材
//创建 MpNews 消息
WxMpArticle article1=new WxMpArticle("CSDN 博客","1wSwU1MJFZI-6YasolX_
sB4yNKVM5doGcI8cEQBy6L4yQvYBbOHga_NUTiJQxsIiIYydEQPlMNE_07-XD3wz_Q","牟云飞",
    "http://blog.csdn.net/myfmyfmyfmyf",
    "测试发送新闻信息。。<br/><font color='red'>支持 html 标签</font>。。。
<br/><a href='http://blog.csdn.net/myfmyfmyfmyf'> 欢迎查看博客 http://blog.
csdn.net/myfmyfmyfmyf</a>",
    "描述信息","1");
List<WxMpArticle> list = new ArrayList<WxMpArticle>();
list.add(article1);
WxMpNewsSource news = new WxMpNewsSource("0",list);
//推送 MpNews 消息
System.out.println(JSONObject.fromObject(news).toString());
//第一个参数 json 消息体，第二个参数 url 链接
wxUtil.sendReqMsg(JSONObject.fromObject(news).toString(),"https://qyapi.
weixin.qq.com/cgi-bin/material/add_mpnews?access_token=");
```

执行成功后将返回如下信息：

```
{ "errcode":0,"errmsg":"ok","media_id":"2BH_L-P-Szw11RthX3ruXNvJ971E-e-s4
JYVKvStmT7iiJTN9-1TdSy2iYdDa0H-B"}
```

也可以通过企业号管理端（<https://qy.weixin.qq.com/>）查看，单击【消息中心】|【素材库】|【图文】，如图 3.12 所示。



图 3.12 查看永久图文素材

3.6.6 获取永久素材（非图文素材）

通过 media_id 获取上传的图片、语音、文件、视频素材。

(1) Https 请求链接如下：

https://qyapi.weixin.qq.com/cgi-bin/material/get?access_token=ACCESS_TOKEN&media_id=MEDIA_ID&agentid=AGENTID

注意：获取时需要应用 ID（agentId），所以上传时必须添加应用 ID。

(2) 数据请求方式。

GET 方式进行数据请求。

(3) 参数说明。详细说明如表 3.21 所示。

表 3.21 获取永久素材（非图文）请求参数说明

参数	是否必需	说明
access_token	是	调用接口凭证（需做缓存处理）
agentid	是	应用ID
media_id	是	素材文件标识

(4) 权限说明。

只能在素材指定（agentId）的应用中使用。

(5) 返回结果。

返回结果和普通的 http 下载相同，需要 http 头做相应的处理，如文件类型、文件名等。详细代码请参照“3.6.3 获取临时素材文件”：

```
{
  HTTP/1.1 200 OK
  Connection: close
  Content-Type: image/jpeg
  Content-disposition: attachment; filename="MEDIA_ID.jpg"
  Date: Sun, 06 Jan 2013 10:20:18 GMT
  Cache-Control: no-cache, must-revalidate
  Content-Length: 339721

  Xxxx
}
```

错误返回信息如下所示：

```
{
  "errcode": "40004",
  "errmsg": "invalid media_id"
}
```

若拥有该 media_id 的应用不在管理组的可见范围，则返回：

```
{
  "errcode": "60011",
  "errmsg": "no privilege to access/modify contact/party/agent "
}
```

3.6.7 获取永久素材（图文素材）

通过 media_id 获取上传的图片、语音、文件、视频素材。

(1) Https 请求链接如下：

https://qyapi.weixin.qq.com/cgi-bin/material/get?access_token=ACCESS_TOKEN&media_id=MEDIA_ID&agentid=AGENTID

(2) 数据请求方式。

GET 方式进行数据请求。

(3) 参数说明。详细说明如表 3.22 所示。

表 3.22 获取永久素材（图文）请求参数说明

参数	是否必需	说明
access token	是	调用接口凭证（需做缓存处理）
agentid	是	应用ID
media_id	是	素材文件标识

(4) 权限说明。

只能在素材指定（agentId）的应用中使用。

(5) 返回结果：

```
{
  "type": "mpnews",
  "mpnews": {
    "articles": [
      {
        "thumb_media_id":
"2-G6nrLmr5EC3MMb_-zK1dDdzmd0p7cNliYu9V5w7o8K0HuucGBZCzw4HmLa5C",
        "title": "Title01",
        "author": "zs",
        "digest": "airticle01",
        "content_source_url": "",
        "show_cover_pic": 0
      },
      {
        "thumb_media_id":
"2-G6nrLmr5EC3MMb_-zK1dDdzmd0p7cNliYu9V5w7oovsUPf3wG4t9N3tE",
        "title": "Title02",
        "author": "Author001",
        "digest": "article02",

```

```
        "content_source_url": "",
        "show_cover_pic": 0
    }
  ]
}
```

错误返回信息如下所示：

```
{
  "errcode": "40004",
  "errmsg": "invalid media_id"
}
```

若拥有该 media_id 的应用不在管理组的可见范围，则返回：

```
{
  "errcode": "60011",
  "errmsg": "no privilege to access/modify contact/party/agent "
}
```

3.6.8 删除永久素材

通过 media_id 删除上传的永久素材，包括图文、图片、语音、文件、视频素材等。

(1) Https 请求链接如下：

https://qyapi.weixin.qq.com/cgi-bin/material/del?access_token=ACCESS_TOKEN&agentid=AGENTID&media_id=MEDIA_ID

(2) 数据请求方式。

GET 方式进行数据请求。

(3) 参数说明。详细说明如表 3.23 所示。

表 3.23 删除永久素材请求参数说明

参数	是否必需	说明
access_token	是	调用接口凭证（需做缓存处理）
agentid	是	应用ID，整型，可在应用的设置页面查看
media_id	是	素材文件标识

(4) 权限说明。

管理组需要该应用的“管理”权限。只能删除当前应用内的素材，不同应用的相同内容素材，删除时相互之间不影响。

(5) 返回结果：

```
{
  "errcode": 0,
  "errmsg": "deleted"
}
```


3.6.9 修改永久图文素材

对于需要部分修改的永久素材信息，可以通过当前接口修改素材内容，以完善已有素材。

(1) Https 请求链接如下：

https://qyapi.weixin.qq.com/cgi-bin/material/update_mpnews?access_token=ACCESS_TOKEN

(2) 数据请求方式。

POST 方式进行数据请求。

(3) 消息请求结构体：

```
{
  "agentid": 0,
  "media_id":
  "2MKloSBkGMNTs_kXxuBIzjZA_a9GdD66rdelZYAZVYhaMeBMImiDzlv84HOwy5wqsYZTXZcy_HV
wJ3iZzPgIYNw",
  "mpnews": {
    "articles": [
      {
        "title": "Title01",
        "thumb_media_id":
        "2CQQkmXPbHWxZnyLG3Y3ZgSnafR040HI45myZ6dTGvAhchyAEg5dHKYfnLXn5-2ngCrYUggL32v
t_tfCUjHlsLA",
        "author": "zs",
        "content_source_url": "",
        "content": "Content001",
        "digest": "airticle01",
        "show_cover_pic": "0"
      },
      {
        "title": "Title02",
        "thumb_media_id":
        "2CQQkmXPbHWxZnyLG3Y3ZgSnafR040HI45myZ6dTGvAhchyAEg5dHKYfnLXn5-2ngCrYUggL32v
t_tfCUjHlsLA",
        "author": "Author001",
        "content_source_url": "",
        "content": "UpdateContent002",
        "digest": "Updatearticle02",
        "show_cover_pic": "0"
      }
    ]
  }
}
```

参数详细说明如表 3.24 所示。

表 3.24 修改永久图文素材请求参数说明

参数	是否必需	说明
access_token	是	调用接口凭证（需做缓存处理）
agentid	是	企业应用的ID，整型，可在应用的设置页面查看
media_id	是	素材资源标识
articles	是	图文消息，一个图文消息支持1到10个图文
title	是	图文消息的标题
thumb_media_id	是	图文消息缩略图的media_id，可以在上传永久素材接口中获得
author	否	图文消息的作者
content_source_url	否	图文消息点击“阅读原文”之后的页面链接
content	是	图文消息的内容，支持html标签
digest	否	图文消息的描述
show_cover_pic	否	是否显示封面：1为显示，0为不显示。默认为0

(4) 权限说明。

管理组需要该应用的“管理”权限。只能修改当前应用内的素材，不同应用的相同内容素材修改时互不影响。

(5) 返回结果：

```
{
  "errcode": 0,
  "errmsg": "deleted"
}
```

3.6.10 获取素材总数

获取应用素材总数以及每种类型素材的数目。

(1) Https 请求链接如下：

https://qyapi.weixin.qq.com/cgi-bin/material/get_count?access_token=ACCESS_TOKEN&agentid=AGENTID

(2) 数据请求方式。

GET 方式进行数据请求。

(3) 参数说明：详细说明如表 3.25 所示。

表 3.25 获取素材总数请求参数说明

参数	是否必需	说明
access_token	是	调用接口凭证（需做缓存处理）
agentid	是	应用ID，整型，可在应用的设置页面查看

(4) 权限说明。

管理组拥有企业号各应用的使用权限即可。

(5) 返回结果:

```
{
  "errcode": 0,
  "errmsg": "ok",
  "total_count": 37,
  "image_count": 12,
  "voice_count": 10,
  "video_count": 3,
  "file_count": 3,
  "mpnews_count": 6
}
```

参数详细说明如表 3.26 所示。

表 3.26 获取素材总数请求参数说明

参数	说明
errcode	消息执行结果: 0成功, 1失败
errmsg	执行结果说明
total_count	应用素材总数目
image_count	图片素材总数目
voice_count	音频素材总数目
video_count	视频素材总数目
file_count	文件素材总数目
mpnews_count	图文素材总数目

3.6.11 获取素材列表

获取应用素材详细列表信息。

(1) Https 请求链接如下:

https://qyapi.weixin.qq.com/cgi-bin/material/batchget?access_token=ACCESS_TOKEN

(2) 数据请求方式。

POST 方式进行数据请求。

(3) 消息请求结构体:

```
{
  "type": "image",
  "agentid": 1,
  "offset": 0,
  "count": 10
}
```

获取素材列表参数详细说明如表 3.27 所示。

表 3.27 获取素材列表请求参数说明

参数	是否必需	说明
access_token	是	调用接口凭证（需做缓存处理）
type	是	素材类型，可以为图文（mpnews）、图片（image）、音频（voice）、视频（video）或文件（file）
agentid	是	企业应用的ID，整型，可在应用的设置页面查看
offset	是	从该类型素材的该偏移位置开始返回，0表示从第一个素材 返回
count	是	返回素材的数量，取值在1~50之间

(4) 权限说明。

管理组拥有企业号各应用的使用权限即可。

(5) 返回结果：

若为图片、文件、视频、音频，则返回格式如下：

```
{
  "errcode": 0,
  "errmsg": "ok",
  "type": "image",
  "total_count": 12,
  "item_count": 1,
  "itemlist": [
    {
      "media_id": "2qN9QW-6HI3-AXuvAMi0vYQTyAm7k0Vgiuf7t5Kl4hjOwhYGwY",
      "filename": "test01.png",
      "update_time": 1434686658
    }
  ]
}
```

若为永久图文消息素材列表，则返回如下：

```
{
  "errcode": 0,
  "errmsg": "ok",
  "type": "mpnews",
  "total_count": 20,
  "item_count": 3,
  "itemlist": [
    {
      "media_id": "2-G6nrLmr5EC3MMb_-zKldDdzmd0p7cNliYu",
      "articles": [
        {
          "title": "Title01",
          "thumb_media_id": "2-G6nrLmr5EC3MMb_-zKldDdzmd0p7cNliYu9V5w7o8K0",
          "author": "zs",

```

```
        "content_source_url": "",
        "digest": "airticle01",
        "show_cover_pic": "0"
    },
    //可能有多篇文章
],
"update_time": "1380000000"
},
//可能有多个图文消息结构
]
```


参数详细说明如表 3.28 所示。

表 3.28 获取素材总数结果参数说明

参数	说明
type	素材类型，可以为图文（mpnews）、图片（image）、音频（voice）、视频（video）或文件（file）
total_count	应用该类型素材总数目
item_count	返回该类型素材数目
itemlist	返回该类型素材列表
media_id	图文素材的媒体ID
articles	图文消息，一个图文消息支持1到10个图文
title	图文消息的标题
thumb_media_id	图文消息缩略图的media_id，可以在上传多媒体文件接口中获得。此处的thumb_media_id即为上传接口返回的media_id
author	图文消息的作者
content_source_url	图文消息点击“阅读原文”之后的页面链接
digest	图文消息的描述
show_cover_pic	是否显示封面：1为显示，0为不显示

3.6.12 管理端素材维护

企业号管理端提供了可视化的素材维护页面，登录管理端【消息中心】|【素材库】，根据自身需要，选择相应的素材类型，如图 3.13 所示。



注意：管理端创建的为永久消息。通过接口上传的临时素材，在管理端是无法查找的，只能通过接口查找。上传的永久素材则可以在管理端查找到。



图 3.13 管理端素材维护

注意：应根据不同的素材类型。创建不同的素材。每种素材创建内容都有限制，请酌情填写。

3.7 企业号应用管理

主动模式下，除处理发送消息、管理素材之外，对于每一个创建的应用，都可以通过接口设置应用的头像、名称、简介，还可以通过接口开启或关闭应用的功能开关。本节将简要介绍企业号应用的管理接口，在实际开发中，建议采用后台管理端直接维护各应用情况，其原因是：

- （1）由于应用管理接口功能不如管理端功能丰富；
- （2）客户操作应用权限较少，不像消息、素材等操作频繁；
- （3）能够减少代码的书写。

3.7.1 获取企业号应用

可以通过该 API 接口获取企业号某个应用的基本信息，包括头像、昵称、账号类型、认证类型、可见范围等。

（1）Https 请求链接如下：

```
https://qyapi.weixin.qq.com/cgi-bin/agent/get?access_token=ACCESS_TOKEN&agentid=AGENTID
```

（2）数据请求方式。

GET 方式进行数据请求。

（3）参数说明如表 3.29 所示。

表 3.29 获取企业号应用请求参数说明

参数	是否必需	说明
access_token	是	调用接口凭证（需做缓存处理）
agentid	是	企业应用的ID，整型，可在应用的设置页面查看

(4) 权限说明。

管理组拥有企业号当前应用的发送消息权限。

(5) 返回结果：

```
{
  "errcode": "0",
  "errmsg": "ok",
  "agentid": "1",
  "name": "NAME",
  "square_logo_url": "xxxxxxx",
  "round_logo_url": "yyyyyyyy",
  "description": "desc",
  "allow_userinfos": {
    "user": [
      {
        "userid": "id1",
        "status": "1"
      },
      {
        "userid": "id2",
        "status": "1"
      },
      {
        "userid": "id3",
        "status": "1"
      }
    ]
  },
  "allow_partys": {
    "partyid": [1]
  },
  "allow_tags": {
    "tagid": [1,2,3]
  },
  "close": 0,
  "redirect_domain": "www.qq.com",
  "report_location_flag": 0,
  "isreportuser": 0,
  "isreportenter": 0,
  "type": 1
}
```

参数详细说明如表 3.30 所示。

表 3.30 获取企业号应用结果参数说明

参数	说明
agentid	企业应用ID
name	企业应用名称
square_logo_url	企业应用方形头像
round_logo_url	企业应用圆形头像
description	企业应用详情
allow_userinfos	企业应用可见范围（人员），其中包括userid和关注状态state
allow_partys	企业应用可见范围（部门）
allow_tags	企业应用可见范围（标签）
close	企业应用是否被禁用
redirect_domain	企业应用可信域名
report_location_flag	企业应用是否打开地理位置上报：0 不上报；1 进入会话上报；2 持续上报
isreportuser	是否接收用户变更通知：0 不接收；1 接收
isreportenter	是否上报用户进入应用事件：0 不接收；1 接收
type	应用类型：1 消息型；2 主页型

3.7.2 设置企业号应用

可以通过该 API 接口设置企业应用的选项设置信息，如地理位置上报等。

注意：第三方服务商不能调用该接口设置授权的主页型应用。

- (1) Https 请求链接如下：
`https://qyapi.weixin.qq.com/cgi-bin/agent/set?access_token=ACCESS_TOKEN`
- (2) 数据请求方式。
POST 方式进行数据请求。
- (3) 消息请求结构体：

```
{
  "agentid": "5",
  "report_location_flag": "0",
  "logo_mediaid": "xxxxx",
  "name": "NAME",
  "description": "DESC",
  "redirect_domain": "xxxxxxx",
  "isreportuser": 0,
  "isreportenter": 0,
  "home_url": "http://www.qq.com"
}
```

参数详细说明如表 3.31 所示。

表 3.31 设置企业号应用请求参数说明

参数	是否必需	说明
access_token	是	调用接口凭证（需做缓存处理）
agentid	是	企业应用的ID，整型，可在应用的设置页面查看
access_token	是	调用接口凭证
agentid	是	企业应用的ID
report_location_flag	是	企业应用是否打开地理位置上报：0 不上报；1 进入会话上报；2 持续上报
logo_mediaid	是	企业应用头像的mediaid，通过多媒体接口上传图片获得mediaid，上传后会自动裁剪成方形和圆形两个头像
name	是	企业应用名称
description	是	企业应用详情
redirect_domain	是	企业应用可信域名
isreportuser	是	是否接收用户变更通知：0 不接收；1 接收
isreportenter	是	是否上报用户进入应用事件：0 不接收；1接收
home_url	是	主页型应用URL。URL必须以http或者https开头。消息型应用无须该参数

（4）权限说明。

管理组拥有企业号当前应用的管理权限。

（5）返回结果：

```
{
  "errcode": "0",
  "errmsg": "ok"
}
```

3.7.3 获取应用概况列表

可以通过该 API 接口获取 Secret 所在管理组内的应用概况，返回管理组内应用的 ID 及名称、头像等信息。

（1）Https 请求链接如下：

https://qyapi.weixin.qq.com/cgi-bin/agent/list?access_token=ACCESS_TOKEN

（2）数据请求方式：

GET 方式进行数据请求。

（3）参数说明如表 3.32 所示。

表 3.32 获取应用概况列表请求参数说明

参数	是否必需	说明
access_token	是	调用接口凭证（需做缓存处理）

（4）权限说明。

管理组拥有的企业号应用。

(5) 返回结果:

```
{
  "errcode": 0,
  "errmsg": "ok",
  "agentlist": [
    {
      "agentid": "5",
      "name": "企业小助手",
      "square_logo_url": "url",
      "round_logo_url": "url"
    },
    {
      "agentid": "8",
      "name": "HR 小助手",
      "square_logo_url": "url",
      "round_logo_url": "url"
    }
  ]
}
```


参数详细说明如表 3.33 所示。

表 3.33 获取应用概况列表请求参数说明

参数	说明
errcode	返回码: 0成功, 1失败
errmsg	返回码的文本描述内容
agentid	应用ID
name	应用名称
square_logo_url	方形头像URL
round_logo_url	圆形头像URL

3.7.4 管理端应用管理

企业号应用的大部分信息操作性较小,通过管理端即可一次性维护。登录企业号管理端【应用中心】,单击相应的应用,即可打开该应用的详细信息,如应用 logo、应用名称、显示模式等,如图 3.14 所示,在这里可以直接对信息进行维护。



注意: 在“模式选择”中,“普通模式”为默认模式,即只有推送信息等简单操作;“回调模式”(将在第 4 章详细讲解)基本可以使用全部接口,而认证后的“回调模式”便可以使用全部接口。



图 3.14 管理端素材维护

3.8 主动模式自定义菜单

主动模式下菜单的创建与维护，只能通过管理端实现。如果通过接口的方式进行维护，则必须开启被动回调模式方可进行菜单接口维护（详细说明请见 4.4 节）。本节将为大家演示主动模式下接口的创建，包括主动模式下菜单的创建、删除以及维护等。

- 01 登录企业号管理后台，然后依次单击【应用中心】|【XX 应用】|【普通模式】|【设置】，打开自定义菜单，如图 3.15 所示。

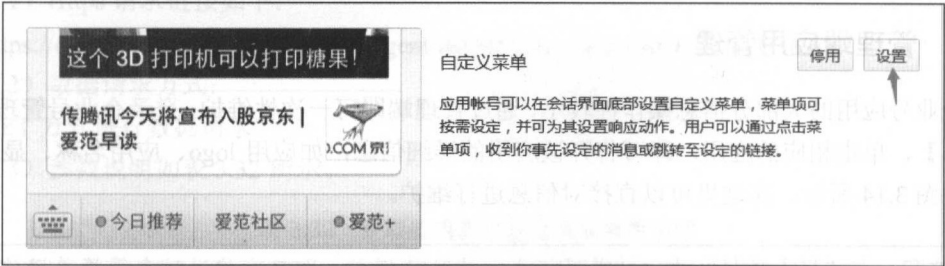


图 3.15 菜单维护页面

- 02 打开自定义菜单管理页面，如图 3.16 所示，左侧为菜单可视化管理区域，右侧为菜单预览区域，可以单击加号（“+”）创建自己的应用菜单吧。这里需要提醒的是，菜单在创建过程中不会更新到企业号中，创建、修改完成后必须单击【发布】，单击发布

之后将在 24 小时内更新至企业号中。



使用技巧：如果需要及时查看更新，则可以先取消关注，然后再重新关注，更新的内容便会出现在菜单中了。

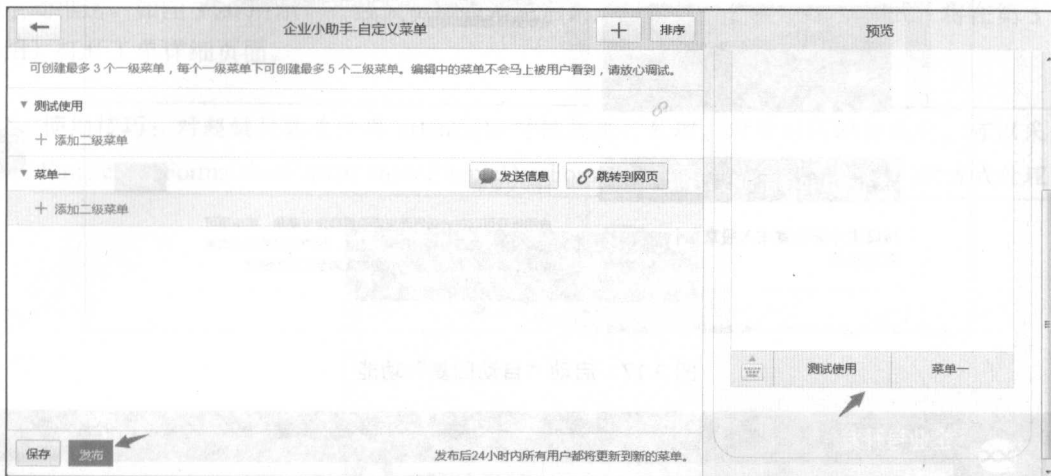


图 3.16 菜单维护页面



注意：每个应用最多可以创建三个一级菜单，每个一级菜单下最多可以创建五个二级菜单。每个一级菜单名称不能多于 4 个汉字或 8 个字母，超出部分将被自动截取。每个二级菜单名称不能多于 8 个汉字或 16 个字母，超出部分将被自动截取。

3.9 信息自动回复

通过此功能可以针对用户的行为，设置特定的文字、语音、图片、视频来作为消息的自动响应回复，当用户符合你所制定的规则时，将会自动回复相应的消息。在主动模式下，此功能只能通过管理端进行管理，需要通过接口响应消息，使用被动回调模式（详细介绍请参照第 4 章）与 Lucene、solr 或数据库等技术相结合实现智能回复。通过对本节的学习，读者可了解并掌握如何通过管理端进行消息规则制定，以及如何实现消息的自动回复。

首先，登录企业号管理后台，然后依次单击【应用中心】|【选择应用】|【普通模式】|【自动回复】|【启用】，启用“自动回复”功能，如图 3.17 所示。

打开消息“自动回复”维护页面，可以根据需要自行创建新的规则，实现“消息自动回复”以及“关键字自动回复”等，如图 3.18 所示。



图 3.17 启动“自动回复”功能



图 3.18 “自动回复”功能

微信端效果显示如图 3.19 所示。



图 3.19 “自动回复”效果图

3.10 案例：业务派单

业务派单是较为常见的业务场景，由系统自动或客服人工等方式，向一线业务人员派发工单，提高信息的时效性和业务办理效率。在技术实现上，其实是 Text 消息或 News 等其他消息的主动推送。如图 3.20 所示，一线人员接收消息后，通过链接（借助 JSAPI 模式，将在第 5 章介绍）打开工单详细页面。

 **使用技巧：**对超链接需要使用 urlencode 对链接进行处理。对于时间的格式化，可以采用 SimpleDateFormat dateFormat=new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")的方式处理。

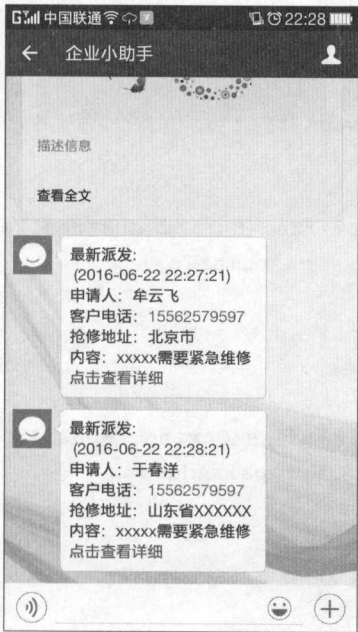


图 3.20 业务派单

示例代码如下：

01 建立工单类 Business.java:

```
package myf.caption3.demo3_5;
/**
 * @author 牟云飞
 * 工单信息表
 */
public class Business {

    private String bizId;
    private String customerName;
    private String customerTel;
    private String repairAddress;
```

```
private String context;
//添加其他属性

public String getBizId() {
    return bizId;
}
public void setBizId(String bizId) {
    this.bizId = bizId;
}
public String getCustomerName() {
    return customerName;
}
public void setCustomerName(String customerName) {
    this.customerName = customerName;
}
public String getCustomerTel() {
    return customerTel;
}
public void setCustomerTel(String customerTel) {
    this.customerTel = customerTel;
}
public String getRepairAddress() {
    return repairAddress;
}
public void setRepairAddress(String repairAddress) {
    this.repairAddress = repairAddress;
}
public String getContext() {
    return context;
}
public void setContext(String context) {
    this.context = context;
}
public Business(String bizId, String customerName, String customerTel,
    String repairAddress, String context) {
    super();
    this.bizId = bizId;
    this.customerName = customerName;
    this.customerTel = customerTel;
    this.repairAddress = repairAddress;
    this.context = context;
}
}
```

02 建立派发工单方法 dispatchBusi2Wx():

```

package myf.caption3.demo3_5;
import java.text.SimpleDateFormat;
import java.util.Date;
import net.sf.json.JSONObject;

/**
 * @author 牟云飞
 * 派发工单
 */
public class Demo3_3_8 {
    //调试方法
    public static void main(String[] args) {
        Business busi = new Business("2016062215", "牟云飞", "1556257xxxx", "
北京市", "xxxxxx 需要紧急维修");
        Demo3_3_8 demo3_3_8 = new Demo3_3_8();
        demo3_3_8.dispatchBusi2Wx(busi, null, "muyunfei", null);
    }
    /**
     * 派发工单方法
     * @param biz 工单类
     * @param toTag 接受人所属标签
     * @param toUser 接受人
     * @param toParty 接受部门
     */
    public JSONObject dispatchBusi2Wx(Business busi,String toTag,String
toUser,String toParty){
        //使用 urlencode 对链接进行处理
        String
old_url=WxUtil.webUrl+"/demoAction.do?action=showDetail&busiId="+busi.getBiz
Id();

        old_url=old_url.replaceAll("/", "%2f").replaceAll(":", "%3a").replaceAll
("\\?", "%3f").replaceAll("=", "%3d").replaceAll("\\&", "%26");
        //System.out.println(old_url);
        String url="https://open.weixin.qq.com/connect/oauth2/authorize?
appid="+WxUtil.MESSAGE_CORPID+"&redirect_uri="
+old_url
+"&response_type=code&scope=snsapi_base&state=oaNews#wechat_redirect";
        //推送时间
        SimpleDateFormat dateFormat =new SimpleDateFormat("yyyy-MM-dd HH:mm:
ss");
        String timeString=dateFormat.format(new Date());
        //生成推送内容
        String content="最新派发:\r\n"
+ " (" +timeString+")\r\n"
+"申请人: "+(busi.getCustomerName()==null?"":busi.getCustomerName())
+"\r\n"

```



```

        +"客户电话: "+(busi.getCustomerTel()==null?"":busi.getCustomerTel())
        +"\r\n"
        +(busi.getRepairAddress()==null?"":("抢修地址: "
        +busi.getRepairAddress()))+"\r\n")
        +"内容: "+(busi.getContext()==null?"":busi.getContext())+"\r\n"
        +"<a href='"+url+"'>点击查看详细</a>";
        //3.5.2 节中已展示 Text 消息源码
        WxTextMessage txtMessage=new WxTextMessage(toUser, toParty, toTag,
        "text", "0", content, "0");
        WxUtil util=new WxUtil();
        //发送内容
        JSONObject result = util.sendReqMsg((ReqBaseMsg)txtMessage);
        System.out.println(result.toString());
        return result;
    }
}

```



注意：对 URL 进行 urlencode 处理，是为了能够进行 OAuth 身份验证，避免因链接暴露而引起非法访问问题，提高信息安全，详细说明将在第 8 章介绍。

第 4 章

被动回调模式

主动调用模式仅能实现企业号主动推送接口，不能实现消息接收等，而在回调模式下，企业不仅可以主动调用企业号接口，还可以接收成员的消息或事件。通过对本章的学习，使读者将掌握如何配置回调模式，如何接收、解析、使用回调消息等。

本章主要涉及的知识有：

- 概念被动回调模式：学会被动回调模式基础知识，了解模式注意事项。
- 模式消息的加密/解密：学会如何加密、解密回调模式消息。
- 自定义菜单的管理：学会如何创建、管理菜单。
- 各类消息体的接收/解析：掌握各类消息的信息结构，学会如何接收、解析消息。
- 接收事件：学会如何接收、使用各类事件。
- 被动响应消息：学会如何响应接收到的消息、事件，掌握各类被动响应消息的消息体结构以及发送方式等。
- 业务与接口的实际应用：通过本章最后的示例，学习回调模式下接口如何使用，如何通过本章所学的知识，灵活开发微信企业号。

4.1 被动回调模式介绍

主动调用模式是企业号向员工推送消息，而被动回调模式（也叫被动模式、回调模式）则是在主动调用的基础上，增加了接收员工向企业号发送的消息。在企业号开发过程中，配置最麻烦、注意最多的也是被动回调模式，就像将一箱苹果分配给人很容易，如果让人自己来挑选则不易。被动回调模式就像那箱苹果一样，需要满足不同客户的需要，需要兼顾各种类型的事件、链接以及安全性能等，所以在配置上相对烦琐，被动回调模式信息传递流程如图 4.1 所示。

注意：URL 验证是在每次传递消息时都进行验证，但 GET 中验证只验证一次，即配置回调链接时进行验证。

在接受消息上，回调模式先通过配置链接，以 GET 形式发送一个密文，我们需要在 GET 中解析密文，并将 echostr 返回给微信，完成回调 URL 的验证，URL 验证成功之后（URL 验证

仅在修改回调链接时需要），微信将以 POST 形式将加密的真正内容发送过来，如图 4.1 所示。

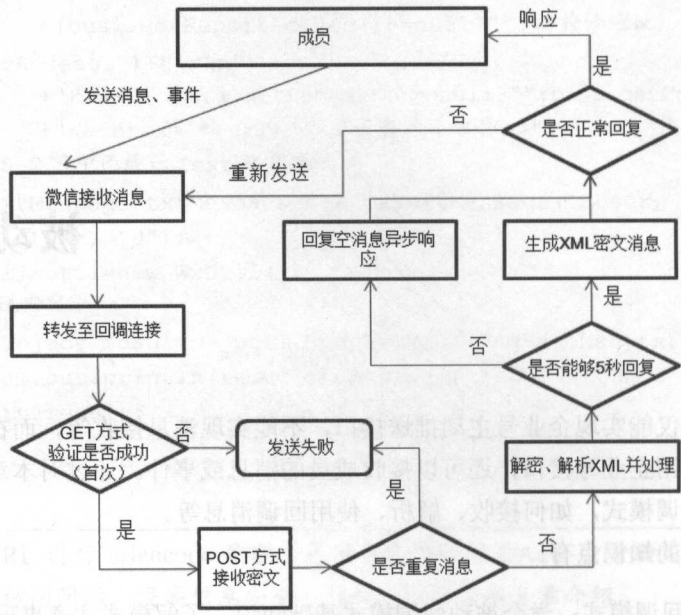


图 4.1 被动回调模式消息传递流程

回调模式在配置上还需要注意以下几点。

- 接收的信息为 XML 形式，并非 JSON 格式。
- 必须具有一个外网域名（回调模式下可以不是 ICP 备案域名，任意外网域名即可），建议读者使用 ICP 备案的域名，因为在之后的“第 5 章 JSAPI 模式”中，必须使用 ICP 备案的域名。
- EncodeAESKey 生成规则是 32 位明文经过 Base64 加密后，去掉“=”，形成的 43 位密钥。EncodeAESKey 建议手动生成，不要采用随机生成的，早期官网提供的随机生成是不能使用的，有时成功，有时失败，目前暂无测试。
- JDK 版本必须是 1.6 版本或以上。
- 根据相应的 JDK，替换相应的 JCE 安全策略补丁包，补丁完成之后，重启 Java 虚拟机服务。
- 消息接受之前需要 GET 验证，并且接收的消息是 UTF-8 编码的、加密的 XML 格式消息，需要解密之后方能使用。
- 回调模式和主动调用模式在消息发送上也有很大不同：
- A 回调模式下，被动发送的消息必须是 XML 格式并进行加密，加密规则是，首先进行 AES 加密，然后进行 Base64 加密。
- B 主动发送消息，格式为 JSON 格式，不需要加密，但需要有效的 AccessToken。
- 回调模式接收到真正的消息内容之后，应在五秒钟内回复。对于五秒钟内无法做出响应的，也应回复空消息，使用异步接口进行回复。对于无须响应的消息，回复空消息即可。空消息微信将不做信息提示。

- 微信服务器在五秒钟内收不到响应会断掉连接，并重新发起请求，总共重试三次。如果在调试中，发现成员无法收到响应的消息，可以检查是否是消息处理超时。当接收成功后，http 头部返回 200，表示接收成功，其他错误码一律当作失败并发起重试。关于重试的消息排重，有 msgid 的消息推荐使用 msgid 排重，事件类型消息推荐使用 FromUserName + CreateTime 排重。

注意：被动模式包含主动模式的推送接口，所以绝大多数应用都将开启回调模式（主页型信息除外）。

4.2 开启回调模式

每个企业号应用都有两个模式可以选择，默认是“普通模式”，需要手动修改为“回调模式”，如图 4.2 所示。普通模式下只能进行消息的主动推送、简单的自动回复以及菜单管理（“普通模式”不能调用接口进行维护）。

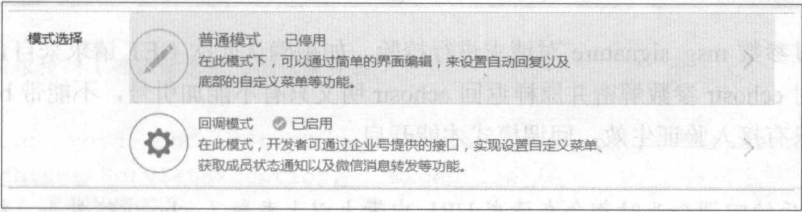


图 4.2 模式选择

开启应用的“回调模式”后，将要求填写应用的 URL、Token 和 EncodingAESKey 三个参数，如图 4.3 所示。URL 是企业号应用接收企业号推送请求的访问协议和地址，支持 HTTP 或 HTTPS 协议；Token 可由企业任意填写，用于生成签名；EncodingAESKey 用于消息体的加密，是 AES 密钥的 Base64 编码。这里的 EncodingAESKey 建议参照 4.1 节介绍的，手动生成。

企业小助手-回调模式

开启回调模式，请先填写接口配置信息
请填写接口配置信息，此信息需要你拥有自己的服务器资源。
填写的URL需要正确响应微信验证 URL 的请求，具体说明请阅读接口文档。

URL
以http://或https://开头

Token
英文或数字，长度为3-32字符 随机获取

EncodingAESKey
英文或数字，长度为43字符 随机获取

保存

图 4.3 开启回调模式

注意：这里的 Token 与主动调用的 AccessToken 不一样。消息接收下的 Token 是固定的，用于加密、解密消息体，而消息推送下的 Token 是有有效期的，用于推送各类消息。

单击“保存”按钮，进行 URL 有效性验证，验证时，企业号发送 GET 请求到填写的 URL 上，GET 请求携带四个参数 msg_signature、timestamp、nonce 和 echostr，详细说明如表 4.1 所示，企业在获取时需要做 urldecode 处理，否则会验证不成功。

表 4.1 上传临时素材请求参数说明

参数	是否必需	说明
msg_signature	是	微信加密签名，msg_signature结合了企业填写的Token、请求中的timestamp、nonce参数、加密的消息体
timestamp	是	时间戳
nonce	是	随机数
echostr	首次校验必带	加密的随机字符串，以msg_encrypt格式提供。需要解密并返回echostr明文，解密后有random、msg_len、msg、\$CorpID四个字段，其中msg即为echostr明文

企业通过参数 msg_signature 对请求进行校验，如果确认此次 GET 请求来自企业号，那么企业应用会对 echostr 参数解密并原样返回 echostr 明文只有不能加引号，不能带 bom 头，不能带换行符，只有接入验证生效，回调模式才能开启。

注意：后续回调企业时都会在请求 URL 中带上以上参数（echostr 除外），校验方式与首次验证 URL 一致。

示例代码如下。

01 这里使用普通的 Servlet 进行演示，修改 web.xml 增加 Servlet 配置：

```
<!-- 确认微信服务器的请求类，回调模式使用-->
<servlet>
    <servlet-name>coreServlet</servlet-name>
    <servlet-class>
        myf.caption4.servlet.CoreServlet
    </servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>coreServlet</servlet-name>
    <url-pattern>/coreServlet</url-pattern>
</servlet-mapping>
```

02 创建回调类 CoreServlet.java，并增加 doGet、doPost 方法，在 doGet 中完成 UL 的验证：

```
package myf.caption4.servlet;

import java.io.IOException;
```



```

import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import myf.caption4.servlet.QQTool.WXBizMsgCrypt;
import myf.caption4.servlet.service.CoreService;
import myf.caption4.servlet.WxUtil;

/**
 * 请求处理的核心类，回调模式
 */
public class CoreServlet extends HttpServlet {
    private static final long serialVersionUID = 4440739483644821986L;

    /**
     * 请求校验（确认请求来自微信服务器）
     */
    public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // 微信加密签名
        String signature = request.getParameter("msg_signature");
        System.out.println("signature:"+signature);
        // 时间戳
        String timestamp = request.getParameter("timestamp");
        System.out.println("timestamp:"+timestamp);
        // 随机数
        String nonce = request.getParameter("nonce");
        System.out.println("nonce:"+nonce);
        // 随机字符串
        String echostr = request.getParameter("echostr");
        System.out.println("echostr:"+echostr);
        String sToken = WxUtil.RESP_MESSAGE_TOKEN;
        String sCorpID = WxUtil.RESP_MESSAGE_CORPID;
        String sEncodingAESKey = WxUtil.RESP_MESSAGE_ENCODINGAESKEY;
        try {
            WXBizMsgCrypt wxcpt = new WXBizMsgCrypt(sToken, sEncodingAESKey,
sCorpID);

            String sEchoStr; //需要返回的明文
            sEchoStr = wxcpt.VerifyURL(signature, timestamp,
                nonce, echostr);
            System.out.println("verifyurl echostr: " + sEchoStr);
            // 验证 URL 成功，将 sEchoStr 返回

```

```
        PrintWriter out = response.getWriter();
        out.write(sEchoStr);
        out.flush();
        out.close();
    } catch (Exception e) {
        //验证 URL 失败，错误原因请查看异常
        e.printStackTrace();
    }
}

/**
 * 处理微信服务器发来的消息
 */
public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    //读取消息，执行消息处理
    CoreService.processRequest(request, response);
}
}
```

注意：加密、解密方法将在 4.3 节中介绍。做过服务号开发的读者需要注意，企业号开发的 GET 验证中是 msg_signature、timestamp、nonce、echostr 四个参数进行排序解密而服务号开发则是 signature、timestamp、nonce 三个参数进行排序解密。

完成 URL 验证之后，回调模式就正式开启了。进入“回调模式”页面，可以自定义菜单（可以通过接口维护）、是否开启状态变动通知、是否上报地理位置、是否上报进入应用事件、是否支持微信消息转发等，如图 4.4 所示。

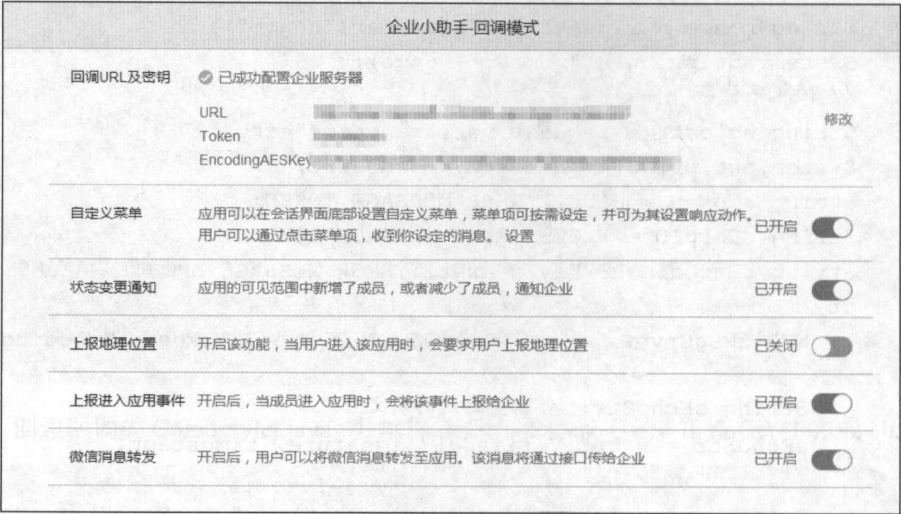


图 4.4 “回调模式” 页面

4.3 加密/解密算法

加密、解密是回调模式中特有的功能，接收消息时，微信将以 POST 形式发送密文消息，需要将密文进行解密，获得明文消息体，并在五秒钟内完成消息的处理。对于需要响应的消息，需要进行加密方可回复；对于不需要响应的消息，回复空字符串即可。这里需要注意，回调消息必须进行回复。对于消息体的加密、解密详细说明如下。

(1) 术语及其说明，详细说明如下。

- `msg_signature` 是签名，用于验证调用者的合法性。
- `EncodingAESKey` 用于消息体加密，长度固定为 43 个字符，从 a~z、A~Z、0~9 共 62 个字符中选取，是 AESKey 的 Base64 编码。解码后即为 32 字节长的 AESKey。
- `AESKey=Base64_Decode(EncodingAESKey + "=")`，是 AES 算法的密钥，长度为 32 字节。AES 采用 CBC 模式，数据采用 PKCS#7 填充至 32 字节的倍数；IV 初始向量大小为 16 字节，取 AESKey 前 16 字节；
- `msg` 为消息体明文，格式为 XML。
- `msg_encrypt = Base64_Encode(AES_Encrypt[random(16B) + msg_len(4B) + msg + $CorpID])`，是对明文消息 `msg` 加密处理后的 Base64 编码。其中 `random` 为 16 字节的随机字符串；`msg_len` 为 4 字节的 `msg` 长度，网络字节序；`msg` 为消息体明文；`$CorpID` 为企业号的标识。

(2) GET 验证 URL。

为了验证调用者的合法性，微信在回调 URL 中增加了消息签名，以参数 `msg_signature` 标识，企业需要验证此参数的正确性后再解密。验证步骤如下：

- 01 企业计算签名 `dev_msg_signature=sha1(sort(token、timestamp、nonce、msg_encrypt))`，`sort` 的含义是将参数按照字母字典排序，然后从小到大拼接成一个字符串。

注意：有服务号开发经验的读者需要注意，GET 验证 URL 时，这里的参数是 `msg_signatur`，而不是 `signature`。

- 02 比较 `dev_msg_signature` 和 `msg_signature` 是否相等，相等则表示验证通过。

- 03 返回 `.echostr`。

注意：消息接受的 URL 验证与回调配置的 URL 验证方式相同。

(3) POST 消息体解密。

消息内容的解密主要分为以下几步。

- 01 对密文 BASE64 解码 `aes_msg=Base64_Decode(msg_encrypt)`。
- 02 使用 AESKey 做 AES 解密 `rand_msg=AES_Decrypt(aes_msg)`。
- 03 验证解密后的 `$CorpID`、`msg_len`。

04 去掉 rand_msg 头部的 16 个随机字节、4 个字节的 msg_len 和尾部的 \$CorpID，即为最终的消息体原文 msg。

05 获得 XML 格式的消息内容后进行提取操作。

(4) POST 消息响应加密。

消息响应是以 XML 格式进行发送，其中 Encrypt 为加密的密文消息，XML 格式如下：

```
<xml>
  <Encrypt><![CDATA[msg_encrypt]]></Encrypt>
  <MsgSignature><![CDATA[msg_signature]]></MsgSignature>
  <TimeStamp>timestamp</TimeStamp>
  <Nonce><![CDATA[nonce]]></Nonce>
</xml>
```

对明文 msg 加密的过程如下：

msg_encrypt = Base64_Encode(AES_Encrypt(random(16B) + msg_len(4B) + msg + \$CorpID)), AES 加密的 buf 由 16 个字节的随机字符串、4 个字节的 msg 长度、明文 msg 和 \$CorpID 组成，其中 msg_len 为 msg 的字节数（网络字节序），\$CorpID 为企业号的 CorpID。经 AESKey 加密后，再进行 Base64 编码，即，获得密文 msg_encrypt。

(5) 下载地址及返回码。

微信向我们提供了各个版本的加密、解密工具类，其中，WXBizMsgCrypt.java 类封装了 VerifyURL、DecryptMsg 和 EncryptMsg 三个接口，分别用于开发者验证回调 URL、接收消息的解密以及开发者回复消息的加密过程，下载地址（Java 版）如下：

<http://download.csdn.net/detail/myfmyfmyfmyf/9559465>



注意：JDK 版本至少在 1.6 及以上，并需要对 JDK 进行 JCE 补丁修改，详细说明见 2.1.3 节。

示例代码如下：

```
//初始化函数
WXBizMsgCrypt wxcpt = new WXBizMsgCrypt(sToken, sEncodingAESKey, sCorpID);
//验证 URL, VerifyURL 函数
String sEchoStr; //需要返回的明文
    sEchoStr = wxcpt.VerifyURL(signature, timestamp, nonce, echostr);
//密文解密, DecryptMsg 函数
    String sMsg = wxcpt.DecryptMsg(sReqMsgSig, sReqTimeStamp, sReqNonce,
sReqData);
    System.out.println("after decrypt msg: " + sMsg); //输出解密后的文件
//消息响应, 消息加密, EncryptMsg 函数
String sRespData=WxUtil.messageToXml(txtMsg); //XML 格式响应消息
    String sEncryptMsg = wxcpt.EncryptMsg(sRespData, time, sReqNonce);
```

错误返回码详细说明如表 4.2 所示。

表 4.2 加密解密返回码

返回码	说明
0	请求成功
-40001	签名验证错误
-40002	XML解析失败
-40003	sha加密生成签名失败
-40004	AESKey 非法
-40005	CorpID 校验错误
-40006	AES 加密失败
-40007	AES 解密失败
-40008	解密后得到的buffer非法
-40009	base64加密失败
-40010	base64解密失败
-40011	生成XML失败

注意：不仅消息接受的 URL 验证与回调配置的 URL 验证方式相同，而且通讯录管理中的异步任务回调链接，也是采用同样的验证调用方式。

4.4 被动模式自定义菜单

企业号的每个消息型应用都可以拥有自己的菜单，开启“回调模式”之后可以调用接口来创建、删除、获取应用菜单。对于“普通模式”，只能人工手动维护，无法通过接口进行操作（详细介绍参见 3.8 节）。通过对本节的学习，读者将掌握如何通过接口创建、删除、获取应用菜单。

4.4.1 限制与说明

每个应用最多可以创建 3 个一级菜单，每个一级菜单下最多可以创建 5 个二级菜单。每个一级菜单名称不能多于 4 个汉字或 8 个字母，超出部分将被自动截取。每个二级菜单名称不能多于 8 个汉字或 16 个字母，超出部分将被自动截取。

注意：官方 API 文档写明二级菜单名称最多 7 个汉字，实际上最多是 8 个，与主动模式菜单相同。

自定义菜单接口可实现多种类型的菜单操作，如图 4.5 所示。

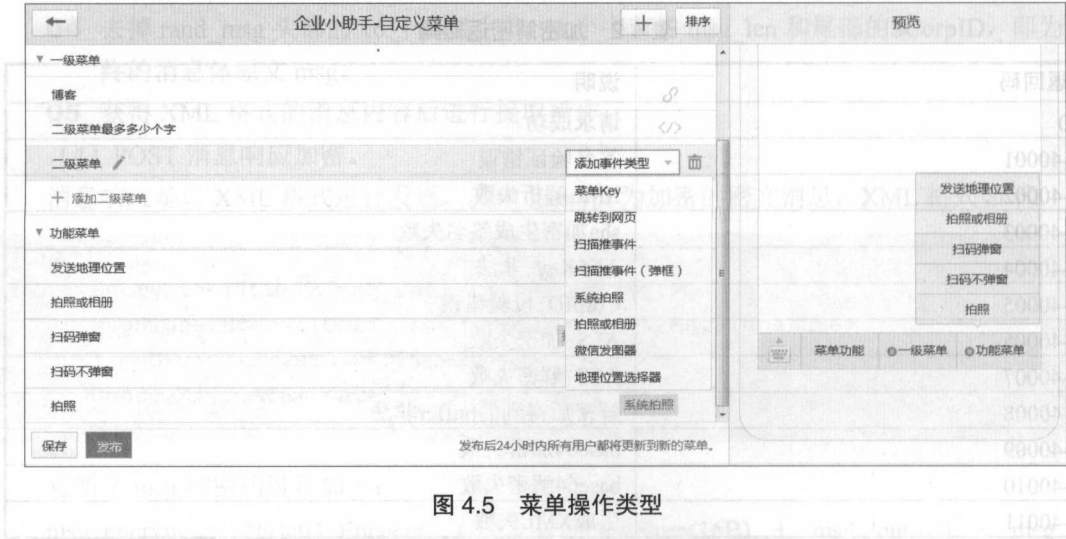


图 4.5 菜单操作类型

菜单操作类型及详细说明如表 4.3 所示。

表 4.3 菜单操作类型及说明

字段值	功能名称	说明
click	点击推事件	成员点击click类型按钮后，微信服务器会通过消息接口推送消息类型为event 的结构给开发者（参考“4.7.3 菜单事件”），并且带上按钮中开发者填写的key值，开发者可以通过自定义的key值与成员进行交互
view	跳转URL	成员点击view类型按钮后，微信客户端将会打开开发者在按钮中填写的网页URL，可与网页授权获取成员基本信息接口结合，获得成员基本信息
scancode_push	扫码推事件	成员点击按钮后，微信客户端将调用“扫一扫”工具，完成扫码操作后显示扫描结果（如果是URL，将进入URL），且会将扫码的结果传给开发者，开发者可以下发消息
scancode_waitmsg	扫码推事件且弹出“消息接收中”提示框	成员点击按钮后，微信客户端将调用“扫一扫”工具，完成扫码操作后，将扫码的结果传给开发者，同时收起“扫一扫”工具，然后弹出“消息接收中”提示框，随后可能会收到开发者下发的消息
pic_sysphoto	弹出系统拍照发图	成员点击按钮后，微信客户端将调用“系统相机”，完成拍照操作后，会将拍摄的照片发送给开发者，并推送事件给开发者，同时收起“系统相机”，随后可能会收到开发者下发的消息
pic_photo_or_album	弹出拍照或者相册发图	成员点击按钮后，微信客户端将弹出选择器供成员选择“拍照”或者“从手机相册选择”。成员选择后即走其他两种流程
pic_weixin	弹出微信相册发图器	成员点击按钮后，微信客户端将调用微信相册，完成选择操作后，将选择的照片发送给开发者的服务器，并推送事件给开发者，同时收起相册，随后可能会收到开发者下发的消息
location_select	弹出地理位置选择器	成员点击按钮后，微信客户端将调用“地理位置选择”工具，完成选择操作后，将选择的地理位置发送给开发者的服务器，同时收起“地理位置选择”工具，随后可能会收到开发者下发的消息



注意：除 click 和 view 外的所有事件，仅支持微信 iPhone 5.4.1/Android 5.4 以上版本，旧版本微信成员点击后将没有回应，开发者也不能正常接收到事件推送。

4.4.2 创建菜单

创建应用菜单详细说明如下。

(1) Https 请求链接：

https://qyapi.weixin.qq.com/cgi-bin/menu/create?access_token=ACCESS_TOKEN&agentid=AGENTID

(2) 数据请求方式。

POST 方式进行数据请求。

(3) 消息请求结构体。创建菜单的请求接口根据类型不同分为以下两种：click 和 view 类型菜单数据请求结构，消息请求结构如下：

```
{
  "button": [
    {
      "type": "click",
      "name": "今日歌曲",
      "key": "V1001_TODAY_MUSIC"
    },
    {
      "name": "菜单",
      "sub_button": [
        {
          "type": "view",
          "name": "搜索",
          "url": "http://www.soso.com/"
        },
        {
          "type": "click",
          "name": "赞一下我们",
          "key": "V1001_GOOD"
        }
      ]
    }
  ]
}
```

扫码、发送图片、发送位置等其他新增按钮类型的菜单数据请求结构，消息请求结构体如下：

```
{
  "button": [
    {
```

```
"name": "扫码",
"sub_button": [
  {
    "type": "scancode_waitmsg",
    "name": "扫码带提示",
    "key": "rselfmenu_0_0",
    "sub_button": [ ]
  },
  {
    "type": "scancode_push",
    "name": "扫码推事件",
    "key": "rselfmenu_0_1",
    "sub_button": [ ]
  }
],
{
  "name": "发图",
  "sub_button": [
    {
      "type": "pic_sysphoto",
      "name": "系统拍照发图",
      "key": "rselfmenu_1_0",
      "sub_button": [ ]
    },
    {
      "type": "pic_photo_or_album",
      "name": "拍照或者相册发图",
      "key": "rselfmenu_1_1",
      "sub_button": [ ]
    },
    {
      "type": "pic_weixin",
      "name": "微信相册发图",
      "key": "rselfmenu_1_2",
      "sub_button": [ ]
    }
  ]
},
{
  "name": "发送位置",
  "type": "location_select",
  "key": "rselfmenu_2_0"
}
]
```

参数详细说明如表 4.4 所示。

表 4.4 创建菜单请求参数说明

参数	是否必需	说明
access_token	是	调用接口凭证，需要缓存
agentid	是	企业应用的ID，整型。可在应用的设置页面查看
button	是	一级菜单数组，个数应为1~3个
sub_button	否	二级菜单数组，个数应为1~5个
type	是	菜单的响应动作类型
name	是	菜单标题，不得超过16个字节
key	click等点击类型必须	菜单KEY值，用于消息接口推送，不超过128字节
url	view类型必须	网页链接，成员点击菜单可打开链接，不超过256字节

注意：这里的 access_tokem 不是回调模式中填写的 Token，而是主动模式下需要缓存的 AccessToken。

(4) 权限说明。

管理组必须拥有当前应用的管理权限，并且应用必须开启“回调模式”，“普通模式”下无法调用接口 API。

(5) 返回参数说明：

```
{
  "errcode":0,
  "errmsg":"ok"
}
```

详细说明如表 4.5 所示。

表 4.5 创建菜单请求参数说明

参数	说明
errcode	执行结果：0成功，1失败
errmsg	执行接口说明

(6) 示例代码如下：

```
public static void main(String[] args) {

    StringBuffer createMenuStr=new StringBuffer("");
    createMenuStr.append("{}");
    createMenuStr.append("{\"button\":[");
    createMenuStr.append("                ");
    createMenuStr.append("                {\"type\": \"click\",");
    //一级菜单只能显示四个汉字
    createMenuStr.append("                {\"name\": \"菜单功能\",");
    createMenuStr.append("                {\"key\": \"TION\"}");
}
```



```

        createMenuStr.append("        },");
        createMenuStr.append("        {");
        createMenuStr.append("            \"name\": \"一级菜单\",");
        createMenuStr.append("            \"sub_button\": [");
        createMenuStr.append("                {");
        createMenuStr.append("                    \"type\": \"view\",");
        createMenuStr.append("                    \"name\": \"博客\",");
        createMenuStr.append("                    \"url\": \"http://blog.csdn.net/myfmyfmyfmyf\"");
        createMenuStr.append("                },");
        createMenuStr.append("            {");
        createMenuStr.append("                \"type\": \"click\",");
        createMenuStr.append("                \"name\": \"二级菜单最多多少个字\",");
        createMenuStr.append("                \"key\": \"KEY1\"");
        createMenuStr.append("            }");
        createMenuStr.append("        ]");
        createMenuStr.append("    }");
        createMenuStr.append("    ]");
        createMenuStr.append("}");
        //创建菜单
        createMenu(createMenuStr.toString(), "0");
    }
    /**
     * 创建菜单
     * @param menu
     * @return
     */
    public static boolean createMenu(String menuStr, String agentId){
        boolean flag=false;
        //获得缓存 token
        //String token=getTokenFromWx();
        String token="Hg3bbHRnVK2Hpq1JoFXm3_KYUdxWyIaimGFVICotm83ghA3SVUB-MgSoxqlH0thI"; //测试 token
        try {
            CloseableHttpClient httpClient = HttpClients.createDefault();
            HttpPost httpPost= new HttpPost("https://qyapi.weixin.qq.com/cgi-bin/menu/create?access_token="+token+"&agentid="+ agentId);
            //设置请求实体
            StringEntity myEntity = new StringEntity(menuStr,
                ContentType.create("text/plain", "UTF-8"));
            httpPost.setEntity(myEntity);
            //Create a custom response handler
            ResponseHandler<JSONObject> responseHandler = new ResponseHandler<JSONObject>() {
                public JSONObject handleResponse(

```



```

        final HttpResponse response) throws ClientProtocolException,
IOException {
    int status = response.getStatusLine().getStatusCode();
    if (status >= 200 && status < 300) {
        //得到响应数据
        HttpEntity entity = response.getEntity();
        if (null != entity) {
            String result = EntityUtils.toString(entity);
            //根据字符串生成 JSON 对象
            JSONObject resultObj = JSONObject.fromObject
(result);

            return resultObj;
        } else {
            return null;
        }
    } else {
        throw new ClientProtocolException("Unexpected response
status: " + status);
    }
}

};
//返回的 JSON 对象
JSONObject responseBody = httpClient.execute(httpPost, responseHandler);
System.out.println(responseBody);
httpClient.close();
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
return flag;
}

```

注意：对于使用超过限制的菜单名，将提示{"errcode":40018,"errmsg":"invalid button name size"}错误。

菜单创建效果图如图 4.6 所示。

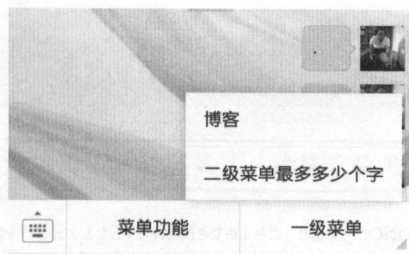


图 4.6 创建菜单



注意：细心的读者可能发现了，二级菜单超出字数限制（一级菜单最多 4 个汉字，二级菜单最多 8 个汉字），这是目前微信 API 接口存在的 Bug，通过接口二级菜单可以维护更多的汉字，而通过管理端则只能维护规定的 8 个汉字，不过，建议还是使用 8 个汉字，以免微信后期修复导致界面异常。

在此功能中还有另外一个隐藏问题，则是如果链接中 agentid 填写错误也能够执行创建，默认修改企业号中的第一个应用的菜单。

4.4.3 删除菜单

事务都是相对立的，菜单既然开放了创建接口，那么必定也有删除接口。删除应用菜单详细说明如下。

(1) Https 请求链接：

https://qyapi.weixin.qq.com/cgi-bin/menu/delete?access_token=ACCESS_TOKEN&agentid=AGENTID

(2) 数据请求方式。

GET 方式进行数据请求。

(3) 参数说明。详细说明如表 4.6 所示。

表 4.6 获取素材总数请求参数说明

参数	是否必须	说明
access token	是	调用接口凭证（需做缓存处理）
agentid	是	应用ID，整型，可在应用的设置页面查看

(4) 权限说明。

管理组必须拥有当前应用的管理权限，并且应用必须开启“回调模式”，“普通模式”下无法调用接口 API。

(5) 返回参数说明：

```
{
  "errcode":0,
  "errmsg":"ok"
}
```

(6) 示例代码：

```
/**
 * 删除菜单
 * @param code
 * @return
 */
public static JSONObject deleteMenu(String agentId){
    //获取缓存 token
    //String token=getTokenFromWx();
```

```

String
token="Hg3bbHRnVK2Hpq1JoFXm3_KYUdxWyIaimGFVICotm83ghA3SVUB-MgSoxqlH0thI";//
测试 token

try {
    CloseableHttpClient httpClient = HttpClients.createDefault();
    HttpPost httpPost= new HttpPost("https://qyapi.weixin.qq.com/
cgi-bin/menu/delete?access_token="+token+"&agentid="+agentId);
    //创建自定义响应处理程序
    ResponseHandler<JSONObject> responseHandler = new
ResponseHandler<JSONObject>() {
        public JSONObject handleResponse(
            final HttpResponse response) throws ClientProtocolException,
IOException {
            int status = response.getStatusLine().getStatusCode();
            if (status >= 200 && status < 300) {
                HttpEntity entity = response.getEntity();
                if(null!=entity){
                    String result= EntityUtils.toString(entity);
                    //根据字符串生成 JSON 对象
                    JSONObject resultObj = JSONObject.fromObject
result);
                    return resultObj;
                }else{
                    return null;
                }
            } else {
                throw new ClientProtocolException("Unexpected status:
" + status);
            }
        }
    };
    //返回的 JSON 对象
    JSONObject responseBody = httpClient.execute(httpPost, responseHandler);
    System.out.println(responseBody);
    return responseBody;
} catch (Exception e) {
    //e.printStackTrace();
    return null;
}
}

```



注意：此功能将会删除该应用下的所有菜单，请慎重使用。

4.4.4 获取菜单列表

获取应用菜单列表的详细说明如下。

(1) Https 请求链接:

https://qyapi.weixin.qq.com/cgi-bin/menu/get?access_token=ACCESS_TOKEN&agentid=AGENTID

(2) 数据请求方式。

GET 方式进行数据请求。

(3) 参数说明。详细说明如表 4.7 所示。

表 4.7 获取素材总数请求参数说明

参数	是否必须	说明
access_token	是	调用接口凭证（需做缓存处理）
agentid	是	应用ID，整型，可在应用的设置页面查看

(4) 权限说明。

管理组必须拥有当前应用的管理权限，并且应用必须开启“回调模式”，“普通模式”下无法调用接口 API。

(5) 返回参数说明：

返回参数与“创建菜单”的请求参数一致，详情参见 4.4.2 节消息请求结构体。

4.4.5 管理端菜单维护

被动回调模式下菜单的创建与维护，不仅能够通过 API 进行管理，而且能够通过管理端进行菜单的创建、删除以及维护，本节将为读者演示被动回调模式下菜单的管理，介绍被动回调模式下菜单的创建、删除以及维护等。

登录企业号管理后台，然后依次单击【应用中心】|【选择应用】|【回调模式】|【自定义菜单】|【启用】|【设置】，打开自定义菜单，如图 4.7 所示。



图 4.7 回调模式管理端菜单管理

读者可以根据业务需要创建相应的菜单，创建完成之后，注意单击“发布”按钮，发布成功之后将在 24 小时内显示菜单。如果需要进行立即测试，则可以清除微信缓存或者重新关注企业号即可。

4.5 接收消息 Dom 解析

通过前面几节的讲解，我们知道企业号在信息接收上采用的是加密的 XML 形式，所以在实现消息处理时，不仅需要进行数据解密处理，还需要解析 XML 文件，本节将讲解 Java Dom 解析的方式。

Java DOM (Document Object Model, 文档对象模型) 是基于 W3C 标准的 XML 解析包，是解析 XML 的方式之一，其他还有 SAX、JDom、Dom4j 等方式。首先通过加载整个文档和构造层次结构，从而形成类似树的数据结构，使其具有一定的层次结构，方便数据提取，接下来了解 W3C DOM 的基础知识。

(1) 解析器工厂类 (DocumentBuilderFactory) :

```
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
```

(2) 创建解析器 (DocumentBuilder) :

```
//通过解析器工厂类来获得
```

```
DocumentBuilder db = dbf.newDocumentBuilder();
```

(3) 获得文档模型 (Document) :

获得文档模型的一种方式是通过 XML 文件的形式，例如：

```
Document doc = db.parse("bean.xml");
```

另一种则是将需要解析的 XML 文档、字符串等转化成 InputStream 输入流，也是接下来我们将要使用的方式，例如：

```
InputStream is = new FileInputStream("bean.xml");  
Document doc = db.parse(is);
```

Document 对象代表了一个 XML 文档的模型树，所有的其他 Node 都以一定的顺序包含在 Document 对象之内，排列成一个树状结构。



注意：Document 对象生成之后，对 XML 文档的所有操作都与解析器无关。

(4) 获得根节点 root，在之后的数据操作中，必须以根节点为开始：

```
//获得根节点数据
```

```
Element root = document.getDocumentElement();
```

(5) 获得节点数据：

```
//获得 MsgType 节点数据
```



```
NodeList nodelist_msgType = root.getElementsByTagName("MsgType");
String recieveMsgType = nodelist_msgType.item(0).getTextContent();
```



使用技巧：获得节点之后，使用 `getTextContent()` 获得节点文本信息，以免使用对象进行操作。

消息接收流程是“解密→解析 XML→加密→响应回复”，示例代码如下：

```
/**
 * 接收消息并响应消息
 * @param request
 * @return xml
 */
public static String processRequest(HttpServletRequest request,
HttpServletResponse response)
    //微信加密签名
    String sReqMsgSig = request.getParameter("msg_signature");
    //时间戳
    String sReqTimeStamp = request.getParameter("timestamp");
    //随机数
    String sReqNonce = request.getParameter("nonce");
    //企业号配置参数
    String sToken = WxUtil.RESP_MESSAGE_TOKEN;
    String sCorpID = WxUtil.MESSAGE_CORPID;
    String sEncodingAESKey = WxUtil.RESP_MESSAGE_ENCODINGAESKEY;
    try {
        //接受信息，通过输入流获得密文数据
        ServletInputStream in = request.getInputStream();
        BufferedReader reader = new BufferedReader(new InputStreamReader(in));
        String sReqData="";
        String itemStr=""; //作为输出字符串的临时串，用于判断是否读取完毕
        while (null!=(itemStr=reader.readLine())) {
            sReqData+=itemStr;
        }
        //对密文消息进行处理获得明文
        WXBizMsgCrypt wxcpt = new WXBizMsgCrypt(sToken, sEncodingAESKey,
sCorpID);
        String sMsg = wxcpt.DecryptMsg(sReqMsgSig, sReqTimeStamp, sReqNonce,
sReqData);
        //输出解密后的文件
        //System.out.println("after decrypt msg: " + sMsg);
        //解析出明文 XML 标签的内容进行处理
        //通过解析器工厂类获得 DOM XML 解析器
        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
        DocumentBuilder db = dbf.newDocumentBuilder();
        //将明文信息存入数据流，获得 Document 元素
```

```

StringReader sr = new StringReader(sMsg);
InputSource is = new InputSource(sr);
Document document = db.parse(is);
//获得根节点数据
Element root = document.getDocumentElement();
//获得某一节点数据, 判断类型
NodeList nodelist_msgType = root.getElementsByTagName("MsgType");
String recieveMsgType = nodelist_msgType.item(0).getTextContent();
String content="";
if("text".equals(recieveMsgType)){//如果是文本消息
    //消息处理
}else if("location".equals(recieveMsgType)){//如果是位置消息
    //消息处理
}else if("event".equals(recieveMsgType)){//如果是时间消息
    //消息处理
}
//!!!!!!!!!!!!!!!!!!!!!!设置回复!!!!!!!!!!!!!!
//-----
//回复人
NodeList nodelist_fromUser = root.getElementsByTagName
("FromUserName");
String mycreate = nodelist_fromUser.item(0).getTextContent();
//回复人
//时间
String time=new Date().getTime()+"";
//应用id
String AgentID="0";
//消息类型
String msg_type="text";
//content="被动响应消息:"+content;
content="";
//生成一个被动响应的消息
TextMessage txtMsg= new TextMessage();
txtMsg.setContent(content);//文字内容
txtMsg.setCreateTime(Long.valueOf(time));//创建时间
txtMsg.setFromUserName(sCorpID);//消息来源
txtMsg.setMsgType(WxUtil.RESP_MESSAGE_TYPE_TEXT);//消息类型
txtMsg.setToUserName(mycreate);
String sRespData=WxUtil.messageToXml(txtMsg);
String sEncryptMsg = wxcpt.EncryptMsg(sRespData, time, sReqNonce);
//输出
PrintWriter out = response.getWriter();
out.write(sEncryptMsg);
out.flush();
out.close();

```

```

    } catch (Exception e) {
        //TODO
        //解密失败，失败原因请查看异常
        e.printStackTrace();
    }
    return "";
}

```



备注：消息的加密、解密数据包说明及下载详见“4.3 加密/解密算法”。messageToXml方法将类转换成XML，以便进行消息加密、回复。

4.6 消息响应 Xstream 转换

在4.5节我们学习了如何进行信息的接收，接下来将要学习的是如何进行信息的响应处理。为了方便在程序中使用面向对象的编程思维，我们将各类响应消息封装成对象，使用Xstream类库返回特定的XML结构字符串。

Xstream是一种OXMapping技术，可以轻易地将Java对象和XML文档相互转换，实现对象的序列化，产生XML结构的数据，而且可以修改某个特定的属性和节点名称。示例代码如下所示：

```

/**
 * 扩展Xstream使其支持CDATA
 * 内部类XppDriver
 */
private static XStream xstream = new XStream(new XppDriver() {
    public HierarchicalStreamWriter createWriter(Writer out) {
        return new PrettyPrintWriter(out) {
            //对所有XML节点的转换都增加CDATA标记
            boolean cdata = true;
            @SuppressWarnings("unchecked")
            public void startNode(String name, Class clazz) {
                super.startNode(name, clazz);
            }
            protected void writeText(QuickWriter writer, String text) {
                if (cdata) {
                    writer.write("<![CDATA[");
                    writer.write(text);
                    writer.write("]]>");
                } else {
                    writer.write(text);
                }
            }
        };
    }
}

```

```

    }
});

/**
 * 文本消息对象转换成 XML
 * @param textMessage 文本消息对象
 * @return xml
 */
public static String messageToXml(TextMessage textMessage) {
    xstream.alias("xml", textMessage.getClass());
    return xstream.toXML(textMessage);
}

/**
 * 图片消息对象转换成 XML
 * @param imageMessage 图片消息对象
 * @return xml
 */
public static String messageToXml(ImageMessage imageMessage) {
    xstream.alias("xml", imageMessage.getClass());
    return xstream.toXML(imageMessage);
}

/**
 * 语音消息对象转换成 XML
 * @param voiceMessage 语音消息对象
 * @return xml
 */
public static String messageToXml(VoiceMessage voiceMessage) {
    xstream.alias("xml", voiceMessage.getClass());
    return xstream.toXML(voiceMessage);
}

/**
 * 视频消息对象转换成 XML
 * @param videoMessage 视频消息对象
 * @return xml
 */
public static String messageToXml(VideoMessage videoMessage) {
    xstream.alias("xml", videoMessage.getClass());
    return xstream.toXML(videoMessage);
}

/**
 * 音乐消息对象转换成 XML

```



```

    * @param musicMessage 音乐消息对象
    * @return xml
    */
    public static String messageToXml(MusicMessage musicMessage) {
        xstream.alias("xml", musicMessage.getClass());
        return xstream.toXML(musicMessage);
    }

    /**
    * 图文消息对象转换成 XML
    * @param newsMessage 图文消息对象
    * @return xml
    */
    public static String messageToXml(NewsMessage newsMessage) {
        xstream.alias("xml", newsMessage.getClass());
        xstream.alias("item", new Article().getClass());
        return xstream.toXML(newsMessage);
    }
}

```



备注：XStream 是 Thoughtworks 的包。CDATA 指的是不由 XML 解析器进行解析的文本数据，标准格式：![CDATA[文本内容]]>，可以解决&、<、>、“、’等特殊字符问题。有服务号开发经验的读者需要注意，响应消息的 FromUserName 是 CorpID，服务号分为 AppID 和微信开发者微信号，服务号响应消息的 FromUserName 是开发者微信号。

通过面向对象的多态性实现消息的回复，示例代码如下：

```

//生成一个被动响应的消息，文本消息
TextMessage txtMsg= new TextMessage();
txtMsg.setContent(content);//文字内容
txtMsg.setCreateTime(Long.valueOf(time));//创建时间
txtMsg.setFromUserName(sCorpID);//消息来源
txtMsg.setMsgType(WxUtil.RESP_MESSAGE_TYPE_TEXT);//消息类型
txtMsg.setToUserName(mycreate);
//通过 Xstream 实现对象到 XML 的转换
String sRespData=WxUtil.messageToXml(txtMsg);
//对 XML 格式的字符串进行加密，参见 4.3 节
String sEncryptMsg = wxcpt.EncryptMsg(sRespData, time, sReqNonce);
//输出响应信息，XML 格式的加密字符串
PrintWriter out = response.getWriter();
out.write(sEncryptMsg);
//清空输出流
out.flush();
//关闭输出流
out.close();

```


4.7 接收普通消息

被动回调消息主要分为三大块：普通消息、事件消息和响应消息，本节将为读者介绍如何接收、解析各类普通消息。

4.7.1 接口说明

普通消息是指成员向企业号应用发送的消息，包括文本、图片、语音、视频、地理位置等类型，企业号会将普通消息推送到每个应用的回调链接中（4.2 节中配置的信息）。在学习如何接收普通消息之前，我们还是先了解下其基本信息，详细说明如下。

（1）数据链接。

开启回调模式中的 URL 链接。

（2）数据获取方式。

以 POST 方式传递数据，数据获取通过数据流的形式 `request.getInputStream()`。

（3）消息类型。

消息型应用支持文本、图片、语音、视频、文件、图文等消息类型，主页型应用仅支持不超过 20 个汉字的文本消息，如图 4.8 所示。



图 4.8 主页型应用接收普通消息

（4）权限说明。

应用必须开启“回调模式”，信息接收失败最多重新发送三次，需要进行消息排重。

（5）示例代码。

建立回调消息处理类，命名为 `AssistServlet.java`（命名随意），使用 `Servlet` 的方式实现，所以此类继承 `HttpServlet` 类，详细代码如下所示：

```
package com.myf.caption4.demo4_4;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.io.StringReader;
```

```

import java.sql.Timestamp;
import java.util.Date;
import javax.servlet.ServletException;
import javax.servlet.ServletInputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import net.sf.json.JSONObject;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;
import com.WxUtil;
import com.myf.QQTool.WXBizMsgCrypt;
/**
 * 回调连接
 */
public class AssistServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    /**
     * 请求校验（确认请求来自微信服务器）
     */
    public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        //微信加密签名
        String signature = request.getParameter("msg_signature");
        System.out.println("signature:"+signature);
        //时间戳
        String timestamp = request.getParameter("timestamp");
        System.out.println("timestamp:"+timestamp);
        //随机数
        String nonce = request.getParameter("nonce");
        System.out.println("nonce:"+nonce);
        //随机字符串
        String echostr = request.getParameter("echostr");
        System.out.println("echostr:"+echostr);
        String sToken = WxUtil.RESP_MESSAGE_TOKEN;
        String sCorpID = WxUtil.MESSAGE_CORPID;
        String sEncodingAESKey = WxUtil.RESP_MESSAGE_ENCODINGAESKEY;
        try {
            WXBizMsgCrypt wxcpt = new WXBizMsgCrypt(sToken, sEncodingAESKey,
sCorpID);

            String sEchoStr; //需要返回的明文
            sEchoStr = wxcpt.VerifyURL(signature, timestamp,

```

```

        nonce, echostr);
        System.out.println("verifyurl echostr: " + sEchoStr);
        //验证 URL 成功, 将 sEchoStr 返回
        PrintWriter out = response.getWriter();
        out.write(sEchoStr);
        out.flush();
        out.close();
    } catch (Exception e) {
        //验证 URL 失败, 错误原因请查看异常
        e.printStackTrace();
    }
}

/**
 * 处理微信服务器发来的消息
 */
public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    //消息处理
    executeMsg(request, response);
}

/**
 * 接收消息并完成消息响应
 * @param request
 * @return xml
 */
public String executeMsg(HttpServletRequest request, HttpServletResponse
response) {
    //微信加密签名
    String sReqMsgSig = request.getParameter("msg_signature");
    //时间戳
    String sReqTimeStamp = request.getParameter("timestamp");
    //随机数
    String sReqNonce = request.getParameter("nonce");
    String sToken = WxUtil.RESP_MESSAGE_TOKEN;
    String sCorpID = WxUtil.MESSAGE_CORPID;
    String sEncodingAESKey = WxUtil.RESP_MESSAGE_ENCODINGAESKEY;
    try {
        //接受信息, 通过输入流获得密文数据
        ServletInputStream in = request.getInputStream();
        BufferedReader reader = new BufferedReader(new InputStreamReader(in));
        String sReqData = "";
        String itemStr = ""; //作为输出字符串的临时串, 用于判断是否读取完毕
        while (null != (itemStr = reader.readLine())) {

```

```

        sReqData+=itemStr;
    }
    //对密文消息进行处理获得明文
    WXBizMsgCrypt wxcpt = new WXBizMsgCrypt(sToken,
sEncodingAESKey, sCorpID);
    String sMsg = wxcpt.DecryptMsg(sReqMsgSig, sReqTimeStamp,
sReqNonce, sReqData);
    //输出解密后的文件
    //System.out.println("after decrypt msg: " + sMsg);
    //解析出明文 XML 标签的内容进行处理
    //通过解析器工厂类获得 DOM xml 解析器
    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
    DocumentBuilder db = dbf.newDocumentBuilder();
    //将明文信息存入数据流, 获得 Document 元素
    StringReader sr = new StringReader(sMsg);
    InputSource is = new InputSource(sr);
    Document document = db.parse(is);
    //获得根节点数据
    Element root = document.getDocumentElement();
    //获得某一节点数据, 判断类型
    NodeList nodelist_msgType = root.getElementsByTagName("MsgType");
    String recieveMsgType = nodelist_msgType.item(0).getTextContent();
    String content="";
    if("text".equals(recieveMsgType)){//如果是文本消息
        //处理文本消息
    }else if("location".equals(recieveMsgType)){//如果是位置消息

    }else if("image".equals(recieveMsgType)){//如果是图片消息

    }

    //!!!!!!!!!!!!!!!!!!!!!!设置回复!!!!!!!!!!!!!!
    //-----
    //回复人
    NodeList nodelist_fromUser = root.getElementsByTagName
("FromUserName");
    String mycreate = nodelist_fromUser.item(0).getTextContent();
    //回复人
    //时间
    String time=new Date().getTime()+"";
    //应用 id
    String AgentID="0";
    //消息类型
    String msg_type="text";
    //content="被动响应消息:"+content;
    content="";
    //生成一个被动响应的消息

```



```

        TextMessage txtMsg= new TextMessage();
        txtMsg.setContent(content);//文字内容
        txtMsg.setCreateTime(Long.valueOf(time));//创建时间
        txtMsg.setFromUserName(sCorpID);//消息来源
        txtMsg.setMsgType(WxUtil.RESP_MESSAGE_TYPE_TEXT);//消息类型
        txtMsg.setToUserName(mycreate);
        String sRespData=WxUtil.messageToXml(txtMsg);
        String sEncryptMsg = wxcpt.EncryptMsg(sRespData, time,
sReqNonce);

        //输出
        PrintWriter out = response.getWriter();
        out.write(sEncryptMsg);
        out.flush();
        out.close();

    } catch (Exception e) {
        //TODO
        //解密失败,失败原因请查看异常
        e.printStackTrace();
    }
    return "";
}
}

```

AssistServlet 类完成之后,需要配置 Web.xml 文件,代码如下:

```

<servlet>
    <servlet-name>assistServlet</servlet-name>
    <servlet-class>
        com.myf.caption4.demo4_4.AssistServlet
    </servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>assistServlet</servlet-name>
    <url-pattern>/assistServlet.slt</url-pattern>
</servlet-mapping>

```



备注: .slt 无特殊含义,可以随意命名,如果使用 struts1、struts2 的读者不建议使用.do 以及.action,因为 struts1、struts2 有分别对应处理.do、.action 的链接,过滤链接可以通过 struts.action.extension 等方式修改,修改之后读者便可以随意使用。

4.7.2 接收文本消息

接收文本信息,接收信息之后,企业号便可针对接收的内容进行人工智能回复。

(1) text 消息明文结构:


```
<xml>
  <ToUserName><![CDATA[toUser]]></ToUserName>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1348831860</CreateTime>
  <MsgType><![CDATA[text]]></MsgType>
  <Content><![CDATA[this is a test]]></Content>
  <MsgId>1234567890123456</MsgId>
  <AgentID>1</AgentID>
</xml>
```


(2) 消息体详细说明如表 4.8 所示。

表 4.8 text消息接收格式

参数	说明
ToUserName	信息接收人，企业号CorpID
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：text
Content	文本消息内容
MsgId	消息ID，64位整型
AgentID	企业应用的ID，整型。可在应用的设置页面查看

(3) 示例代码：

```
//将明文信息存入数据流，获得 Document 元素
StringReader sr = new StringReader(sMsg);
InputSource is = new InputSource(sr);
Document document = db.parse(is);
//获得根节点数据
Element root = document.getDocumentElement();
//获得内容
NodeList nodelist1 = root.getElementsByTagName("Content");
String content = nodelist1.item(0).getTextContent();
//获得信息发送人
NodeList enter_people_note = root.getElementsByTagName("FromUserName");
String enter_people = enter_people_note.item(0).getTextContent();
```

 备注：信息接收其他信息，如信息接收获取、dom 解析等，请参照 4.5 节，后面不再介绍。

4.7.3 接收图片消息

image 消息为图片消息，属于多媒体素材信息，可以获得图片链接及 mediaId。

(1) image 消息明文结构：

```
<xml>
  <ToUserName><![CDATA[toUser]]></ToUserName>
```

```
<FromUserName><![CDATA[fromUser]]></FromUserName>
<CreateTime>1348831860</CreateTime>
<MsgType><![CDATA[image]]></MsgType>
<PicUrl><![CDATA[this is a url]]></PicUrl>
<MediaId><![CDATA[media_id]]></MediaId>
<MsgId>1234567890123456</MsgId>
<AgentID>1</AgentID>
</xml>
```

(2) 消息体详细说明如表 4.9 所示。

表 4.9 image消息接收格式

参数	说明
ToUserName	信息接收人，企业号CorpID
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：image
PicUrl	图片链接
MediaId	图片媒体文件ID，可以调用获取媒体文件接口拉取数据
MsgId	消息ID，64位整型
AgentID	企业应用的ID，整型。可在应用的设置页面查看

(3) 示例代码：

```
//将明文信息存入数据流，获得 Document 元素
StringReader sr = new StringReader(sMsg);
InputSource is = new InputSource(sr);
Document document = db.parse(is);
//获得根节点数据
Element root = document.getDocumentElement();
//获得 mediaId
NodeList nodelist1 = root.getElementsByTagName("MediaId ");
String mediaId = nodelist1.item(0).getTextContent();
```



备注：对于需要保存的图片，需要及时下载，临时素材文件在微信服务器只能保存 3 天时间，下载方式可以通过 mediaId 进行下载，详细说明参见“3.6 素材管理”。

4.7.4 接收音频消息

voice 消息为音频消息，属于多媒体素材信息，可以获得 mediaId 以及音频格式等信息。

(1) voice 消息明文结构：

```
<xml>
  <ToUserName><![CDATA[toUser]]></ToUserName>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1357290913</CreateTime>
```

```
<MsgType><![CDATA[voice]]></MsgType>
<MediaId><![CDATA[media_id]]></MediaId>
<Format><![CDATA[Format]]></Format>
<MsgId>1234567890123456</MsgId>
<AgentID>1</AgentID>
</xml>
```

(2) 消息体详细说明如表 4.10 所示。

表 4.10 voice消息接收格式

参数	说明
ToUserName	信息接收人，企业号CorpID
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：voice
MediaId	语音媒体文件ID，可以调用获取媒体文件接口拉取数据
Format	语音格式，如amr、speex等
MsgId	消息ID，64位整型
AgentID	企业应用的ID，整型。可在应用的设置页面查看

(3) 示例代码：

```
//将明文信息存入数据流，获得 Document 元素
StringReader sr = new StringReader(sMsg);
InputSource is = new InputSource(sr);
Document document = db.parse(is);
//获得根节点数据
Element root = document.getDocumentElement();
//获得 mediaId
NodeList nodelist1 = root.getElementsByTagName("MediaId ");
String mediaId = nodelist1.item(0).getTextContent();
//获得 format
NodeList nodelist1 = root.getElementsByTagName("Format ");
String format = nodelist1.item(0).getTextContent();
```

4.7.5 接收位置消息

location 消息为位置消息，可以获得经纬度、缩放比例等地图坐标信息。

(1) location 消息明文结构

```
<xml>
  <ToUserName><![CDATA[toUser]]></ToUserName>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1351776360</CreateTime>
  <MsgType><![CDATA[location]]></MsgType>
  <Location_X>23.134521</Location_X>
  <Location_Y>113.358803</Location_Y>
```

```
<Scale>20</Scale>
<Label><![CDATA[位置信息]]></Label>
<MsgId>1234567890123456</MsgId>
<AgentID>1</AgentID>
</xml>
```



注意：开发时 location 是区分大小写的。小写的 location 接收的是地图信息，大写的 LOCATION 则是后面要讲解的接收位置事件，以免混淆。

(2) 消息体详细说明如表 4.11 所示。

表 4.11 location消息接收格式

参数	说明
ToUserName	信息接收人，企业号CorpID
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：location
Location_X	地理位置纬度
Location_Y	地理位置经度
Scale	地图缩放大小
Label	地理位置信息
MsgId	消息ID，64位整型
AgentID	企业应用的ID，整型。可在应用的设置页面查看

(3) 示例代码：

```
//将明文信息存入数据流，获得 Document 元素
StringReader sr = new StringReader(sMsg);
InputSource is = new InputSource(sr);
Document document = db.parse(is);
//获得根节点数据
Element root = document.getDocumentElement();
//获得纬度信息
NodeList nodelist1 = root.getElementsByTagName("Location_X ");
String locationX = nodelist1.item(0).getTextContent();
//获得经度信息
NodeList nodelist1 = root.getElementsByTagName("Location_Y ");
String locationY = nodelist1.item(0).getTextContent();
```



注意：这里获得的地理坐标信息为 GPS 坐标信息，实际使用时需要根据不同的地图坐标进行转换，如腾讯地图坐标、百度地图坐标等均不是 GPS 坐标，使用时要注意转换。

4.7.6 接收小视频消息

shortvideo 消息为小视频消息，属于多媒体素材信息，可以获得 mediaId 以及视频缩略图

ThumbMediaId 等信息。

(1) shortvideo 消息明文结构：

```
<xml>
  <ToUserName><![CDATA[toUser]]></ToUserName>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1357290913</CreateTime>
  <MsgType><![CDATA[shortvideo]]></MsgType>
  <MediaId><![CDATA[media_id]]></MediaId>
  <ThumbMediaId><![CDATA[thumb_media_id]]></ThumbMediaId>
  <MsgId>1234567890123456</MsgId>
  <AgentID>1</AgentID>
</xml>
```

(2) 消息体详细说明如表 4.12 所示。

表 4.12 shortvideo消息接收格式

参数	说明
ToUserName	信息接收人，企业号CorpID
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：shortvideo
MediaId	视频媒体文件ID，可以调用获取媒体文件接口拉取数据
ThumbMediaId	视频消息缩略图的媒体ID，可以调用获取媒体文件接口拉取数据
MsgId	消息ID，64位整型
AgentID	企业应用的ID，整型。可在应用的设置页面查看

(3) 示例代码：

```
//将明文信息存入数据流，获得 Document 元素
StringReader sr = new StringReader(sMsg);
InputSource is = new InputSource(sr);
Document document = db.parse(is);
//获得根节点数据
Element root = document.getDocumentElement();
//获得 MediaId
NodeList nodelist1 = root.getElementsByTagName("MediaId ");
String mediaId = nodelist1.item(0).getTextContent();
//获得 ThumbMediaId
NodeList nodelist1 = root.getElementsByTagName("ThumbMediaId ");
String thumbMediaId = nodelist1.item(0).getTextContent();
```



备注：缩略图为视频的首帧或其他帧图片，对于视频处理感兴趣的读者可以通过 FFmpeg 等工具实现视频转换和缩略图提取。

4.7.7 接收链接消息

link 消息为链接信息，在消息接收上与其他消息类型并无差异，不同之处在于，之前介绍的 text、image、voice、location、shortvideo 等消息的发送均由按键操作，而 link 消息却没有，如图 4.9 所示。这是多余的接口吗？其实不然，link 消息可以通过“收藏”、“转发”等形式发送，所以也需要做相应的消息处理，详细说明如下。



图 4.9 消息界面

(1) link 消息明文结构:

```
<xml>
  <ToUserName><![CDATA[toUser]]></ToUserName>
  <FromUserName><![CDATA[muyunfei]]></FromUserName>
  <CreateTime>1467274108</CreateTime>
  <MsgType><![CDATA[link]]></MsgType>
  <Title><![CDATA[如何打造企业自己的 DNA? ]]></Title>
  <Description><![CDATA[企业号是一个企业文化建设方式]]></Description>
  <Url><![CDATA[http://mp.weixin.qq.com/s?__biz= d]]></Url>
  <PicUrl><![CDATA[this is a url]]></PicUrl>
  <MsgId>1234567890123456</MsgId>
  <AgentID>1</AgentID>
</xml>
```

(2) 消息体详细说明如表 4.13 所示。

表 4.13 shortvideo消息接收格式

参数	说明
ToUserName	信息接收人，企业号CorpID

参数	说明
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：link
Title	标题
Description	描述
Url	link消息的链接地址
PicUrl	封面缩略图的URL
MsgId	消息ID，64位整型
AgentID	企业应用的ID，整型。可在应用的设置页面查看

(3) 示例代码：

```
//将明文信息存入数据流，获得 Document 元素
StringReader sr = new StringReader(sMsg);
InputSource is = new InputSource(sr);
Document document = db.parse(is);
//获得根节点数据
Element root = document.getDocumentElement();
//获得 URL
NodeList nodelist1 = root.getElementsByTagName("Url ");
String url = nodelist1.item(0).getTextContent();
//获得 PicUrl
NodeList nodelist1 = root.getElementsByTagName("PicUrl ");
String picUrl = nodelist1.item(0).getTextContent();
```



备注：对于 link 消息中的链接，可以通过 HttpURLConnection 等方式实现数据抓取，如下所示：

```
//建立链接
HttpURLConnection httpConn = (HttpURLConnection) oaUrl.openConnection();
//获得抓取数据流，操作数据流实现抓取
InputStream in = httpConn.getInputStream();
```

4.7.8 接收视频消息

video 消息为视频信息，在消息体结构及开发上，与 shortvideo 消息类似，但在发送上与 link 消息一致，均是在“我的收藏”中发送，这里要介绍的是 video 消息的接收，说明如下。

(1) video 消息明文结构：

```
<xml>
  <ToUserName><![CDATA[toUser]]></ToUserName>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1357290913</CreateTime>
  <MsgType><![CDATA[video]]></MsgType>
```

```
<MediaId><![CDATA[media_id]]></MediaId>
<ThumbMediaId><![CDATA[thumb_media_id]]></ThumbMediaId>
<MsgId>1234567890123456</MsgId>
<AgentID>1</AgentID>
</xml>
```

(2) 消息体详细说明如表 4.14 所示。

表 4.14 video消息接收格式

参数	说明
ToUserName	信息接收人，企业号CorpID
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：video
MediaId	视频媒体文件ID，可以调用获取媒体文件接口拉取数据
ThumbMediaId	视频消息缩略图的媒体ID，可以调用获取媒体文件接口拉取数据
MsgId	消息ID，64位整型
AgentID	企业应用的ID，整型。可在应用的设置页面查看

(3) 示例代码：

```
//将明文信息存入数据流，获得 Document 元素
StringReader sr = new StringReader(sMsg);
InputSource is = new InputSource(sr);
Document document = db.parse(is);
//获得根节点数据
Element root = document.getDocumentElement();
//获得 MediaId
NodeList nodelist1 = root.getElementsByTagName("MediaId ");
String mediaId = nodelist1.item(0).getTextContent();
//获得 ThumbMediaId
NodeList nodelist1 = root.getElementsByTagName("ThumbMediaId ");
String thumbMediaId = nodelist1.item(0).getTextContent();
```

4.8 接收事件消息

接收消息分为接收普通消息和接收事件消息，在 4.7 节我们学习了接收普通消息，本节将使读者了解各类事件消息的数据结构，掌握如何正确接收、解析事件消息。

4.8.1 接口说明

事件是指成员在企业号中触发的行为，如关注应用、进入应用、点击菜单、发送地理位置等。成员触发企业号的行为操作，企业号会将普通消息推送到每个应用的回调链接中（4.2 节中配置的信息），接收事件消息在接收解析上基本一致，详细说明如下。

(1) 数据链接。

开启回调模式中的 URL 链接。

(2) 数据获取方式。

以 POST 方式传递数据，数据获取通过数据流的形式 `request.getInputStream()`。

(3) 消息类型。

消息型应用支持成员关注、取消关注、上报地理位置、菜单操作等事件类型，主页型应用则仅支持状态变更通知和上报进入应用事件，如图 4.10 所示。

主页型应用 (测试) -回调模式

回调URL及密钥	已成功配置企业服务器	
URL		修改
Token		
EncodingAESKey		
状态变更通知	应用的可见范围中新增了成员，或者减少了成员，通知企业	已开启 <input checked="" type="checkbox"/>
上报进入应用事件	开启后，当成员进入应用时，会将该事件上报给企业	已关闭 <input type="checkbox"/>

图 4.10 主页型应用回调模式

(4) 权限说明。

应用必须开启“回调模式”；信息接收失败最多重新发送三次，且需要进行消息排重；需要“开启”各类所需事件，如图 4.11 所示。

企业小助手-回调模式

回调URL及密钥	已成功配置企业服务器	
URL		修改
Token		
EncodingAESKey		
自定义菜单	应用可以在会话界面底部设置自定义菜单，菜单项可按需设定，并可为其设置响应动作。用户可以通过点击菜单项，收到你设定的消息。	已关闭 <input type="checkbox"/>
状态变更通知	应用的可见范围中新增了成员，或者减少了成员，通知企业	已开启 <input checked="" type="checkbox"/>
上报地理位置	已开启上报地理位置：仅上报一次 修改	已开启 <input checked="" type="checkbox"/>
上报进入应用事件	开启后，当成员进入应用时，会将该事件上报给企业	已开启 <input checked="" type="checkbox"/>
微信消息转发	开启后，用户可以将微信消息转发至应用。该消息将通过接口传给企业	已开启 <input checked="" type="checkbox"/>

图 4.11 “开启”部分事件通知

(5) 示例代码：

```
if("text".equals(recieveMsgType)){//如果是文本消息
    //处理文本消息
```

```

}else if("location".equals(recieveMsgType)){//如果是位置消息
    //处理位置消息
}else if("image".equals(recieveMsgType)){//如果是图片消息
    //处理图片消息
}else if("event".equals(recieveMsgType)){//如果是时间消息
    //获得事件类型
    NodeList nodelist1 = root.getElementsByTagName("Event");
    String event_type = nodelist1.item(0).getTextContent();
    if("enter_agent".equals(event_type)){//进入应用的事件
        //进入的应用
        NodeList enter_app_note = root.getElementsByTagName("AgentID");
        String enter_app = enter_app_note.item(0).getTextContent();
        //哪个员工进入
        NodeList enter_people_note = root.getElementsByTagName
("FromUserName");
        String enter_people = enter_people_note.item(0).getTextContent();
    }else if("LOCATION".equals(event_type)){//上报地理位置
        //大写 LOCATION 为位置事件, location 为位置消息
    }else if("subscribe".equals(event_type)){
        //关注
    }else if("unsubscribe".equals(event_type)){
        //取消关注
    }
}
}

```



备注：接收普通消息与接收事件消息数据获取、密文解析一致，示例代码请参照 4.7.1 节。

4.8.2 接收关注/取消关注事件

在管理端开启“状态变更通知”之后，成员关注和取消关注企业号将通过每个应用在管理端设置的回调链接推送 subscribe 事件和 unsubscribe 事件，以方便做相应处理，如推送“使用帮助”、“欢迎信息”等。企业号初始应用“企业小助手”，在默认情况下能够获得整个企业号的关注情况，可以利用其特点将人员头像等信息同步到个人数据库，实现数据的双向同步。



注意：成员关注之后，同步成员信息能够自动同步到微信号，但目前自动获取成员微信号尚不稳定，部分成员将出现无法自动获得微信号的情况，建议尽量避免自动获取。

(1) 消息明文结构：

```

<xml>
  <ToUserName><![CDATA[toUser]]></ToUserName>
  <FromUserName><![CDATA[UserID]]></FromUserName>
  <CreateTime>1348831860</CreateTime>
  <MsgType><![CDATA[event]]></MsgType>

```

```
<Event><![CDATA[subscribe]]></Event>
<AgentID>1</AgentID>
</xml>
```

(2) 消息体详细说明如表 4.15 所示。

表 4.15 接收关注/取消关注事件消息格式

参数	说明
ToUserName	信息接收人，企业号CorpID
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：event
Event	事件类型，subscribe（订阅）、unsubscribe（取消订阅）
AgentID	企业应用的ID，整型。可在应用的设置页面获取；如果ID为0，则表示是整个企业号的关注/取消关注事件

(3) 示例代码：

```
//将明文信息存入数据流，获得 Document 元素
StringReader sr = new StringReader(sMsg);
InputSource is = new InputSource(sr);
Document document = db.parse(is);
//获得根节点数据
Element root = document.getDocumentElement();
//获得某一节点数据，判断类型
NodeList nodelist_msgType = root.getElementsByTagName("MsgType");
String recieveMsgType = nodelist_msgType.item(0).getTextContent();
if("event".equals(recieveMsgType)){//如果是时间消息
    //获得事件类型
    NodeList nodelist1 = root.getElementsByTagName("Event");
    String event_type = nodelist1.item(0).getTextContent();
    //关注事件
    if("subscribe".equals(event_type)){
        //关注人
        NodeList enter_people_note = root.getElementsByTagName("FromUserName");
        String enter_people = enter_people_note.item(0).getTextContent();
        NodeList agentID_note = root.getElementsByTagName("AgentID");
        String agentID = agentID_note.item(0).getTextContent();
        System.out.println(enter_people+" 关注了应用 "+agentID);
        //根据人员 ID 获得账户信息，将在第 7 章介绍
        WxUtil util = new WxUtil ();
        JSONObject json=util.getPeopleByUserId(enter_people);
    }else if("unsubscribe".equals(event_type)){//取消关注
        //取消关注人
        NodeList enter_people_note = root.getElementsByTagName("FromUserName");
        String enter_people = enter_people_note.item(0).getTextContent();
        NodeList agentID_note = root.getElementsByTagName("AgentID");
```

```
String agentID = agentID_note.item(0).getTextContent();
System.out.println(enter_people+" 取消了关注, 应用 "+agentID);
}
}
```

4.8.3 接收地理位置事件

读者开启企业号某个应用“上报地理位置”之后，该应用就会上报成员地理位置信息。上报情况分两种：成员进入应用时上报一次和成员处于应用中每 5 秒钟上报一次，如图 4.12 所示。

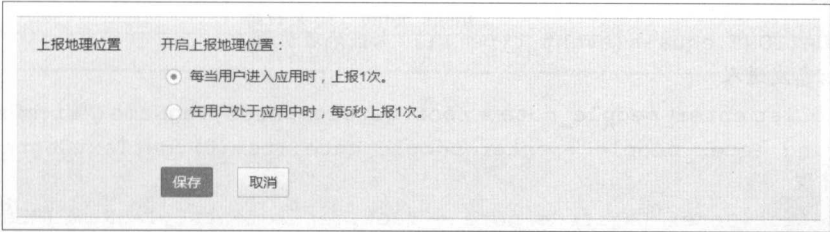


图 4.12 开启“上报地理位置”

备注：开启每 5 秒钟上报一次后，数据增长速度将越来越快，需要根据情况选择是否开启。

(1) 消息明文结构：

```
<xml>
  <ToUserName><![CDATA[toUser]]></ToUserName>
  <FromUserName><![CDATA[FromUser]]></FromUserName>
  <CreateTime>123456789</CreateTime>
  <MsgType><![CDATA[event]]></MsgType>
  <Event><![CDATA[LOCATION]]></Event>
  <Latitude>23.104105</Latitude>
  <Longitude>113.320107</Longitude>
  <Precision>65.000000</Precision>
  <AgentID>1</AgentID>
</xml>
```

注意：位置事件的 Event 为大写的 LOCATION。

(2) 消息体详细说明如表 4.16 所示。

表 4.16 接收地理位置事件消息格式

参数	说明
ToUserName	信息接收人，企业号CorpID
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：event

续表

参数	说明
Event	事件类型，此时固定为：LOCATION
Latitude	地理位置纬度
Longitude	地理位置经度
Precision	地理位置精确度
AgentID	企业应用的ID，整型。可在应用的设置页面查看

(3) 示例代码：

```
if("LOCATION".equals(event_type)){//上报地理位置
    //信息发送人
    NodeList enter_people_note = root.getElementsByTagName("FromUserName");
    String enter_people = enter_people_note.item(0).getTextContent();
    //纬度
    NodeList enter_Latitude_note = root.getElementsByTagName("Latitude");
    String enter_Latitude = enter_Latitude_note.item(0).getTextContent();
    //经度
    NodeList enter_Longitude_note = root.getElementsByTagName("Longitude");
    String enter_Longitude = enter_Longitude_note.item(0).getTextContent();
    //精确度
    NodeList enter_Precision_note = root.getElementsByTagName("Precision");
    String enter_Precision = enter_Precision_note.item(0).getTextContent();
    //设置内容
    content =enter_people+" 进入应用,当前位置: 经度"+enter_Longitude+"纬度
"+enter_Latitude;
}
```

4.8.4 接收进入应用事件

读者开启企业号某个应用的“上报进入应用事件”之后，该应用将会在成员进入时上报进入事件，事件详细说明如下。

(1) 消息明文结构：

```
<xml>
  <ToUserName><![CDATA[toUser]]></ToUserName>
  <FromUserName><![CDATA[muyunfei]]></FromUserName>
  <CreateTime>1467359242</CreateTime>
  <MsgType><![CDATA[event]]></MsgType>
  <AgentID>8</AgentID>
  <Event><![CDATA[enter_agent]]></Event>
  <EventKey><![CDATA[]]></EventKey>
</xml>
```



注意：主页型应用支持上报进入应用事件，官方 API 存有勘误。

(2) 消息体详细说明如表 4.17 所示。

表 4.17 接收进入应用事件消息格式

参数	说明
ToUserName	信息接收人，企业号CorpID
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：event
AgentID	企业应用的ID，整型。可在应用的设置页面查看
Event	事件类型，enter_agent
EventKey	事件KEY值，此事件该值为空

(3) 示例代码：

```
//获得事件类型
NodeList nodelist1 = root.getElementsByTagName("Event");
String event_type = nodelist1.item(0).getTextContent();
if("enter_agent".equals(event_type)){//进入应用的事件
    //进入的应用
    NodeList enter_app_note = root.getElementsByTagName("AgentID");
    String enter_app = enter_app_note.item(0).getTextContent();
    //事件推送人
    NodeList enter_people_note = root.getElementsByTagName("FromUserName");
    String enter_people = enter_people_note.item(0).getTextContent();
    //进入时间
    NodeList enter_CreateTime_note = root.getElementsByTagName("CreateTime");
    String enter_CreateTime = enter_CreateTime_note.item(0).getTextContent();
    //设置响应内容
    content = enter_people+" 进入应用(agentID)为 "+enter_app+" 时间是"+new
Timestamp(Long.valueOf(enter_CreateTime)).toLocaleString();
}
```

4.8.5 接收菜单事件

菜单事件是读者在开启“回调模式”自定义菜单之后，成员触发菜单而产生的信息回调事件，包括扫描、拍照、链接跳转等事件，事件类型如图 4.13 所示。

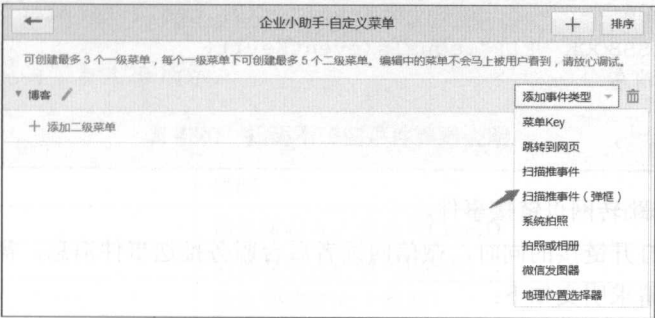


图 4.13 菜单事件

注意：一级菜单弹出二级菜单无触发事件。扫码、拍照及地理位置的菜单事件，仅支持微信 iPhone 5.4.1/Android 5.4 之后版本，之前版本不可用。

接收菜单事件，消息体结构详细说明如下。

(1) 接收菜单拉取消息的事件（又称菜单 KEY 事件）。

成员点击菜单之后，微信向读者后台服务推送 KEY 事件，读者解析密文之后获取 KEY 值，数据请求明文如下：

```
<xml>
<ToUserName><![CDATA[toUser]]></ToUserName>
<FromUserName><![CDATA[FromUser]]></FromUserName>
<CreateTime>123456789</CreateTime>
<MsgType><![CDATA[event]]></MsgType>
<Event><![CDATA[click]]></Event>
<EventKey><![CDATA[EVENTKEY]]></EventKey>
<AgentID>1</AgentID>
</xml>
```

数据参数详细说明如表 4.18 所示。

表 4.18 接收菜单拉取消息的事件参数说明

参数	说明
ToUserName	信息接收人，企业号CorpID
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：event
Event	事件类型，click
EventKey	事件KEY值，与自定义菜单接口中的KEY值对应
AgentID	应用代理ID

KEY 值为自定义值，读者需要根据接收到的 KEY 值进行相应处理，示例代码如下：

```
if("click".equals(event_type)){
    //获得 event_key
    NodeList node_EventKey = root.getElementsByTagName("EventKey");
    String eventKey = node_EventKey.item(0).getTextContent();
    if("ADDRESSBOOK_HELP".equals(eventKey)){
        //回复消息
    }
}
```

(2) 接收菜单跳转网页链接事件。

成员点击菜单打开链接的同时，微信向读者后台服务推送事件消息，需要做记录的业务可以进行记录，数据请求明文如下：

```
<xml>
```

```
<ToUserName><![CDATA[toUser]]></ToUserName>
<FromUserName><![CDATA[FromUser]]></FromUserName>
<CreateTime>123456789</CreateTime>
<MsgType><![CDATA[event]]></MsgType>
<Event><![CDATA[view]]></Event>
<EventKey><![CDATA[www.qq.com]]></EventKey>
<AgentID>1</AgentID>
</xml>
```

数据参数详细说明如表 4.19 所示。

表 4.19 接收菜单跳转页面事件参数说明

参数	说明
ToUserName	信息接收人，企业号CorpID
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：event
Event	事件类型，view
EventKey	事件KEY值，设置跳转URL
AgentID	应用代理ID

（3）扫码事件（不弹窗）。

通过扫描二维码使成员获得获得相应权利或登录某个特权页面等，数据请求明文如下：

```
<xml>
  <ToUserName><![CDATA[toUser]]></ToUserName>
  <FromUserName><![CDATA[FromUser]]></FromUserName>
  <CreateTime>1408090502</CreateTime>
  <MsgType><![CDATA[event]]></MsgType>
  <Event><![CDATA[scancode_push]]></Event>
  <EventKey><![CDATA[6]]></EventKey>
  <ScanCodeInfo>
    <ScanType><![CDATA[qrcode]]></ScanType>
    <ScanResult><![CDATA[1]]></ScanResult>
  </ScanCodeInfo>
  <AgentID>1</AgentID>
</xml>
```


数据参数详细说明如表 4.20 所示。

表 4.20 扫码不弹窗事件参数说明

参数	说明
ToUserName	信息接收人，企业号CorpID
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：event

续表


参数	说明
Event	事件类型，scancode_push
EventKey	事件KEY值，由开发者在创建菜单时设定
ScanCodeInfo	扫描信息
ScanType	扫描类型，一般是qrcode
ScanResult	扫描结果，即二维码对应的字符串信息
AgentID	应用代理ID

 **注意：**扫码事件分为弹窗和不弹窗两种。

(4) 扫码事件（弹窗）。

扫码推事件能够弹出“获取中”提示框，数据请求明文如下：

```
<xml>
  <ToUserName><![CDATA[toUser]]></ToUserName>
  <FromUserName><![CDATA[FromUser]]></FromUserName>
  <CreateTime>1408090606</CreateTime>
  <MsgType><![CDATA[event]]></MsgType>
  <Event><![CDATA[scancode_waitmsg]]></Event>
  <EventKey><![CDATA[6]]></EventKey>
  <ScanCodeInfo>
    <ScanType><![CDATA[qrcode]]></ScanType>
    <ScanResult><![CDATA[2]]></ScanResult>
  </ScanCodeInfo>
  <AgentID>1</AgentID>
</xml>
```

 **备注：**不弹窗扫码事件Event为scancode_push，而弹窗扫码事件Event为scancode_waitmsg。

数据参数详细说明如表 4.21 所示。

表 4.21 扫码弹窗事件参数说明

参数	说明
ToUserName	信息接收人，企业号CorpID
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：event
Event	事件类型，scancode_waitmsg
EventKey	事件KEY值，由开发者在创建菜单时设定
ScanCodeInfo	扫描信息
ScanType	扫描类型，一般是qrcode
ScanResult	扫描结果，即二维码对应的字符串信息
AgentID	应用代理ID

扫码之后，将等待获得消息，效果图如图 4.14 所示。

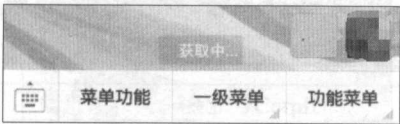


图 4.14 扫码弹窗事件

(5) 地理位置选择器事件。

点击菜单弹出地理位置选择器，选择地理位置后发送至后台服务，数据请求明文如下：

```
<xml>
  <ToUserName><![CDATA[toUser]]></ToUserName>
  <FromUserName><![CDATA[FromUser]]></FromUserName>
  <CreateTime>1408091189</CreateTime>
  <MsgType><![CDATA[event]]></MsgType>
  <Event><![CDATA[location_select]]></Event>
  <EventKey><![CDATA[6]]></EventKey>
  <SendLocationInfo>
    <Location_X><![CDATA[23]]></Location_X>
    <Location_Y><![CDATA[113]]></Location_Y>
    <Scale><![CDATA[15]]></Scale>
    <Label><![CDATA[ 烟台市芝罘区机场路 x 号]]></Label>
    <Poiname><![CDATA[]]></Poiname>
  </SendLocationInfo>
  <AgentID>1</AgentID>
</xml>
```

数据参数详细说明如表 4.22 所示。

表 4.22 地理位置选择器事件参数说明

参数	说明
ToUserName	信息接收人，企业号CorpID
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：event
Event	事件类型，location_select
EventKey	事件KEY值，由开发者在创建菜单时设定
SendLocationInfo	发送的位置信息
Location_X	X坐标信息
Location_Y	Y坐标信息
Scale	精度，可理解为精度或者比例尺，越精细Scale越高
Label	地理位置的字符串信息
Poiname	朋友圈POI的名字，可能为空
AgentID	应用代理ID

点击菜单，显示地理位置选择，如图 4.15 所示。



图 4.15 地理位置选择器事件

(6) 图片发送事件。

通过该接口可以打开拍照或相册发送图片，另外图片发送事件还有单独的拍照事件，以及单独的相册发送事件，详细说明如下。

a. 拍照或相册发送事件：

```
<xml>
  <ToUserName><![CDATA[toUser]]></ToUserName>
  <FromUserName><![CDATA[FromUser]]></FromUserName>
  <CreateTime>1408090816</CreateTime>
  <MsgType><![CDATA[event]]></MsgType>
  <Event><![CDATA[pic_photo_or_album]]></Event>
  <EventKey><![CDATA[6]]></EventKey>
  <SendPicsInfo>
    <Count>1</Count>
    <PicList>
      <item>
        <PicMd5Sum><![CDATA[mediaId]]></PicMd5Sum>
      </item>
    </PicList>
  </SendPicsInfo>
  <AgentID>1</AgentID>
</xml>
```

参数说明如表 4.23 所示。

表 4.23 拍照或相册发送图片事件参数说明

参数	说明
ToUserName	信息接收人，企业号CorpID
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：event
Event	事件类型，pic_photo_or_album
EventKey	事件KEY值，由开发者在创建菜单时设定
SendPicsInfo	发送的图片信息
Count	发送的图片数量
PicList	图片列表
PicMd5Sum	图片的MD5值，开发者若需要，可用于验证接收到的图片
AgentID	应用代理ID

图片选择方式如图 4.16 所示。

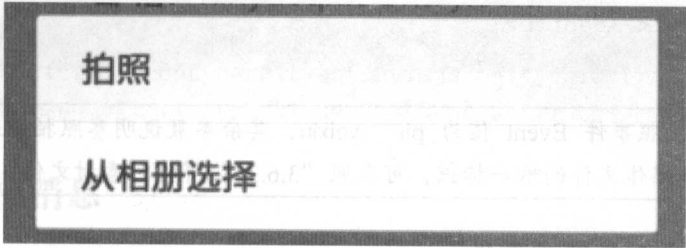


图 4.16 拍照或相册发送图片事件


b. 系统拍照事件。此事件直接弹出拍照界面，无须进行方式选择，事件详细说明如下：

```
<xml>
  <ToUserName><![CDATA[toUser]]></ToUserName>
  <FromUserName><![CDATA[FromUser]]></FromUserName>
  <CreateTime>1408090651</CreateTime>
  <MsgType><![CDATA[event]]></MsgType>
  <Event><![CDATA[pic_sysphoto]]></Event>
  <EventKey><![CDATA[6]]></EventKey>
  <SendPicsInfo><Count>1</Count>
  <PicList>
    <item>
      <PicMd5Sum><![CDATA[mediaId]]></PicMd5Sum>
    </item>
  </PicList>
</SendPicsInfo>
  <AgentID>1</AgentID>
</xml>
```


 **备注：**系统拍照事件 Event 值为 pic_sysphoto，其余参数说明参照拍照或相册发送事件。

c. 相册选择事件：

```
<xml>
  <ToUserName><![CDATA[toUser]]></ToUserName>
  <FromUserName><![CDATA[FromUser]]></FromUserName>
  <CreateTime>1408090816</CreateTime>
  <MsgType><![CDATA[event]]></MsgType>
  <Event><![CDATA[pic_weixin]]></Event>
  <EventKey><![CDATA[6]]></EventKey>
  <SendPicsInfo><Count>1</Count>
  <PicList>
    <item>
      <PicMd5Sum><![CDATA[mediaId]]></PicMd5Sum>
    </item>
  </PicList>
  </SendPicsInfo>
  <AgentID>1</AgentID>
</xml>
```

 **备注：**系统拍照事件 Event 值为 pic_weixin，其余参数说明参照拍照或相册发送事件。
mediaId 为多媒体文件的唯一标识，可参照“3.6.3 获得临时素材文件”获得图片。

4.8.6 接收异步任务完成事件

该事件用于异步任务执行完毕的结果通知，目前只有通讯录批量维护存在异步任务，通讯录异步任务详细说明将在第 7 章介绍，异步任务执行结果事件详细说明如下。

(1) 消息明文结构：

```
<xml>
  <ToUserName><![CDATA[wX28dbb14e37208abe]]></ToUserName>
  <FromUserName><![CDATA[FromUser]]></FromUserName>
  <CreateTime>1425284517</CreateTime>
  <MsgType><![CDATA[event]]></MsgType>
  <Event><![CDATA[batch_job_result]]></Event>
  <BatchJob>
    <JobId><![CDATA[S0MrnndvRG5fadS1LwiBqiDDbM143UqTmKP3152FZk4]]></JobId>
    <JobType><![CDATA[sync_user]]></JobType>
    <ErrCode>0</ErrCode>
    <ErrMsg><![CDATA[ok]]></ErrMsg>
  </BatchJob>
</xml>
```

(2) 消息体详细说明如表 4.24 所示。

表 4.24 接收进入应用事件消息格式

参数	说明
ToUserName	信息接收人，企业号CorpID
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：event
Event	事件的类型，此时固定为batch_job_result
JobId	异步任务id，最大长度为64字符
JobType	操作类型，字符串，目前分别有： 1. sync_user（增量更新成员） 2. replace_user（全量覆盖成员） 3. invite_user（邀请成员关注） 4. replace_party（全量覆盖部门）
ErrCode	返回码
ErrMsg	对返回码的文本描述内容

(3) 示例代码：

```
//获得事件类型
NodeList nodelist1 = root.getElementsByTagName("jobid");
String jobid = nodelist1.item(0).getTextContent();
```

4.9 被动响应消息

读者接收到普通消息或事件消息之后，需要对消息进行回复，回复的消息即为被动响应消息（以下简称响应消息）。前面我们学习了消息的接收，本节将学习如何正确发送响应消息，了解各类响应消息的数据结构，掌握各类响应消息的使用。

4.9.1 接口说明

被动响应消息可以用于快速检索、智能回复等功能，在消息结构上，接收到的消息是加密消息，同样在消息响应时，发送的消息也需要进行加密，详细说明如下。

(1) 数据链接。

开启回调模式中的 URL 链接。

(2) 数据获取方式。

数据获取通过数据流的形式 response.getWriter()，在接收消息之后五秒钟内输出。

(3) 消息类型。

响应消息包括：文字（text）消息、图片（image）消息、声音（voice）消息、视频（video）消息以及图文（news）消息。

(4) 权限说明。

应用必须开启“回调模式”。需在五秒钟内进行回复，否则企业号认为接收失败无响应。

(5) 消息结构体。

消息体包括：msg_signature、timestamp、nonce 以及密文（Encrypt），其中 timestamp 和 nonce 需要读者生成，msg_signature 和密文则是由特定算法生成，消息结构如下。



备注：被动响应消息需要对消息体先进行 AES 加密，处理后再 base64 加密，格式为 XML 格式，算法说明参见 4.3 节。

```
<xml>
  <Encrypt><![CDATA[msg_encrypt]]></Encrypt>
  <MsgSignature><![CDATA[msg_signature]]></MsgSignature>
  <TimeStamp>timestamp</TimeStamp>
  <Nonce><![CDATA[nonce]]></Nonce>
</xml>
```

(6) 示例代码：

```
!!!!!!!!!!!! 回复空消息!!!!!!!!!!!!
//-----
//回复人
NodeList nodelist_fromUser = root.getElementsByTagName("FromUserName");
String mycreate = nodelist_fromUser.item(0).getTextContent();
//时间
String time=new Date().getTime()+"";
//应用 ID
String AgentID="0";
//消息类型
String msg_type="text";
//被动响应消息内容，空消息
content="";
/生成一个被动响应的消息
TextMessage txtMsg= new TextMessage();
txtMsg.setContent(content);//文字内容
txtMsg.setCreateTime(Long.valueOf(time));//创建时间
txtMsg.setFromUserName(sCorpID);//消息来源
txtMsg.setMsgType(WxUtil.RESP_MESSAGE_TYPE_TEXT);//消息类型
txtMsg.setToUserName(mycreate);
//生成响应消息
String sRespData= WxUtil.messageToXml(txtMsg);
String sEncryptMsg = wxcpt.EncryptMsg(sRespData, time, sReqNonce);
//输出
PrintWriter out = response.getWriter();
out.write(sEncryptMsg);
//如果异步响应，也可以回复任何字符串，不会出现提示
//out.write("abcd");
out.flush();
out.close();
```

注意：空消息不会发送提醒，因此成员微信客户端不会做出反应，可以放心回复空消息。企业号中可以回复 content 为空的 消息，也可以直接回复任意字符串，不会像订阅号、服务号一样出现“该公众号暂时无法提供服务，请稍后再试”等任何提示。

4.9.2 被动响应文字消息

被动响应 text 消息为接收消息之后，五秒钟内回复的 text 消息（文字消息），消息详细说明如下：

（1）消息明文结构：

```
<xml>
  <ToUserName><![CDATA[toUser]]></ToUserName>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1348831860</CreateTime>
  <MsgType><![CDATA[text]]></MsgType>
  <Content><![CDATA[this is a test]]></Content>
</xml>
```

备注：空消息也可以是 Content 为空字符串的 text 消息。回复的 text 消息一般有一定的长度显示，建议使用超链接的方式查看详细内容，当文字过多时，显示效果较差。

（2）消息体详细说明如表 4.25 所示。

表 4.25 text 响应消息格式

参数	说明
ToUserName	信息接收人，成员UserID
FromUserName	信息发送人，企业号CorpID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：text
Content	文本消息内容

（3）示例代码。

利用面向对象的方式，封装 text 响应类 RespTextMessage.java，详细代码如下：

```
package myf.caption4.resp;

public class RespTextMessage {
    //接收方账号
    private String ToUserName;
    //开发者微信号
    private String FromUserName;
    //消息创建时间（整型）
    private long CreateTime;
    //消息类型
    private String MsgType;
    //内容
```



```

private String Content;

public String getToUserName() {
    return ToUserName;
}

public void setToUserName(String toUserName) {
    ToUserName = toUserName;
}

public String getFromUserName() {
    return FromUserName;
}

public void setFromUserName(String fromUserName) {
    FromUserName = fromUserName;
}

public long getCreateTime() {
    return CreateTime;
}

public void setCreateTime(long createTime) {
    CreateTime = createTime;
}

public String getMsgType() {
    return MsgType;
}

public void setMsgType(String msgType) {
    MsgType = msgType;
}

public String getContent() {
    return Content;
}

public void setContent(String content) {
    Content = content;
}
}

```

生成 XML 文件并加密响应消息，示例代码如下：

```

//生成一个被动响应的消息
RespTextMessage txtMsg= new RespTextMessage();
txtMsg.setContent(content);//文字内容
txtMsg.setCreateTime(Long.valueOf(time));//创建时间
txtMsg.setFromUserName(sCorpID);//消息来源
txtMsg.setMsgType(WxUtil.RESP_MESSAGE_TYPE_TEXT);//消息类型
txtMsg.setToUserName(mycreate);
//生成 XML 消息
String sRespData=WxUtil.messageToXml(txtMsg);
//加密消息，详细说明参见 4.3 节
String sEncryptMsg = wxcpt.EncryptMsg(sRespData, time, sReqNonce);

```

备注：如果是空消息，直接将 content 设置为""（空字符串）即可。

利用 servlet 输出流直接响应消息，详细代码如下：

```
//输出，response 为 servlet 服务中的 HttpServletResponse
//获得输出打印机
PrintWriter out = response.getWriter();
//输出文件
out.write(sEncryptMsg);
//清空数据流
out.flush();
//关闭数据流
out.close();
```

4.9.3 被动响应图片消息

image 消息为图文消息，接收消息之后，五秒钟内回复的 image 消息为被动响应 image 消息，消息详细说明如下。

(1) 消息明文结构：

```
<xml>
  <ToUserName><![CDATA[toUser]]></ToUserName>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1348831860</CreateTime>
  <MsgType><![CDATA[image]]></MsgType>
  <Image>
    <MediaId><![CDATA[media_id]]></MediaId>
  </Image>
</xml>
```

备注：MediaId 为素材库中的 mediaId，参见 3.6.2 节和 3.6.4 节上传素材图片。

(2) 消息体详细说明如表 4.26 所示。

表 4.26 image响应消息格式

参数	说明
ToUserName	成员UserID
FromUserName	企业号CorpID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：image
MediaId	图片文件ID

(3) 示例代码。

利用面向对象的方式，封装 image 响应类 RespImageMessage.java 和 RespImage.java，详细代码如下：

```
package myf.caption4.resp;

public class RespImageMessage {
    //接收方账号
    private String ToUserName;
    //开发者微信号
    private String FromUserName;
    //消息创建时间（整型）
    private long CreateTime;
    //消息类型
    private String MsgType;
    //图片
    private RespImage Image;
    public String getToUserName() {
        return ToUserName;
    }
    public void setToUserName(String toUserName) {
        ToUserName = toUserName;
    }
    public String getFromUserName() {
        return FromUserName;
    }
    public void setFromUserName(String fromUserName) {
        FromUserName = fromUserName;
    }
    public long getCreateTime() {
        return CreateTime;
    }
    public void setCreateTime(long createTime) {
        CreateTime = createTime;
    }
    public String getMsgType() {
        return MsgType;
    }
    public void setMsgType(String msgType) {
        MsgType = msgType;
    }
    public RespImage getImage() {
        return Image;
    }
    public void setImage(RespImage image) {
        Image = image;
    }
}
```



备注：XStream 的处理方式与 JSON 包处理的方式一样，需要对各对象进行封装。


```
package myf.caption4.resp;
public class RespImage {
    // 媒体文件 ID
    private String MediaId;
    //get、set 方法
    public String getMediaId() {
        return MediaId;
    }
    public void setMediaId(String mediaId) {
        MediaId = mediaId;
    }
}
```

4.9.4 被动响应音频消息

voice 消息为音频消息，接收消息之后，五秒钟内回复的 voice 消息为被动响应 voice 消息，详细说明如下。

(1) 消息明文结构：

```
<xml>
  <ToUserName><![CDATA[toUser]]></ToUserName>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1357290913</CreateTime>
  <MsgType><![CDATA[voice]]></MsgType>
  <Voice>
    <MediaId><![CDATA[media_id]]></MediaId>
  </Voice>
</xml>
```

 备注：MediaId 为素材库中的 mediaId，参照 3.6.2 节和 3.6.4 节中的上传素材文件。

(2) 消息体详细说明如表 4.27 所示。

表 4.27 voice响应消息格式

参数	说明
ToUserName	成员UserID
FromUserName	企业号CorpID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：voice
MediaId	音频文件ID，素材文件ID

(3) 示例代码。

利用面向对象的方式，封装 voice 响应类 RespVoiceMessage.java，详细代码如下：

```
package myf.caption4.resp;
public class RespVoiceMessage {
```



```
//接收方账号
private String ToUserName;
//开发者微信号
private String FromUserName;
//消息创建时间（整型）
private long CreateTime;
//消息类型
private String MsgType;
//图片
private RespMedia Voice;
public String getToUserName() {
    return ToUserName;
}
public void setToUserName(String toUserName) {
    ToUserName = toUserName;
}
public String getFromUserName() {
    return FromUserName;
}
public void setFromUserName(String fromUserName) {
    FromUserName = fromUserName;
}
public long getCreateTime() {
    return CreateTime;
}
public void setCreateTime(long createTime) {
    CreateTime = createTime;
}
public String getMsgType() {
    return MsgType;
}
public void setMsgType(String msgType) {
    MsgType = msgType;
}
public RespMedia getVoice() {
    return Voice;
}
public void setVoice(RespMedia voice) {
    Voice = voice;
}
}
```

备注: RespMedia.java 已在 4.9.3 节中介绍。

4.9.5 被动响应视频消息

video 消息为音频消息，接收消息之后，五秒钟内回复的 video 消息为被动响应 video 消息，详细说明如下。

(1) 消息明文结构：

```
<xml>
  <ToUserName><![CDATA[toUser]]></ToUserName>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1357290913</CreateTime>
  <MsgType><![CDATA[video]]></MsgType>
  <Video>
    <MediaId><![CDATA[media_id]]></MediaId>
    <Title><![CDATA[title]]></Title>
    <Description><![CDATA[description]]></Description>
  </Video>
</xml>
```

备注：MediaId 为素材库中的 mediaId，参照 3.6.2 节和 3.6.4 节中的上传素材文件。

(2) 消息体详细说明如表 4.28 所示。

表 4.28 video响应消息格式

参数	说明
ToUserName	成员UserID
FromUserName	企业号CorpID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：video
MediaId	视频文件ID，可以调用上传媒体文件接口获取
Title	视频消息的标题
Description	视频消息的描述
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：video

(3) 示例代码。

利用面向对象的方式，封装 video 响应类 RespVideoMessage.java 和 RespVideoMedia.java，详细代码如下：

```
package myf.caption4.resp;
public class RespVideoMessage {
    //接收方账号
    private String ToUserName;
    //开发者微信号
    private String FromUserName;
    //消息创建时间(整型)
```

```
private long CreateTime;
//消息类型
private String MsgType;
//视频 media
private RespVideoMedia Video;
public String getToUserName() {
    return ToUserName;
}
public void setToUserName(String toUserName) {
    ToUserName = toUserName;
}
public String getFromUserName() {
    return FromUserName;
}
public void setFromUserName(String fromUserName) {
    FromUserName = fromUserName;
}
public long getCreateTime() {
    return CreateTime;
}
public void setCreateTime(long createTime) {
    CreateTime = createTime;
}
public String getMsgType() {
    return MsgType;
}
public void setMsgType(String msgType) {
    MsgType = msgType;
}
public RespVideoMedia getVideo() {
    return Video;
}
public void setVideo(RespVideoMedia video) {
    Video = video;
}
}

/**
 *视频素材 media
 */
public class RespVideoMedia {
    //媒体文件 ID
    private String MediaId;
    //视频消息的标题
    private String Title;
    //视频消息的描述
```

```

private String Description;
public String getMediaId() {
    return MediaId;
}
public void setMediaId(String mediaId) {
    MediaId = mediaId;
}
public String getTitle() {
    return Title;
}
public void setTitle(String title) {
    Title = title;
}
public String getDescription() {
    return Description;
}
public void setDescription(String description) {
    Description = description;
}
}

```

4.9.6 被动响应图文消息

被动响应 news 消息为五秒钟内能够回复的 news 消息,也是 text 消息之外比较常用的消息,详细说明如下。

(1) 消息明文结构:

```

<xml>
  <ToUserName><![CDATA[toUser]]></ToUserName>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>12345678</CreateTime>
  <MsgType><![CDATA[news]]></MsgType>
  <ArticleCount>2</ArticleCount>
  <Articles>
    <item>
      <Title><![CDATA[title1]]></Title>
      <Description><![CDATA[description1]]></Description>
      <PicUrl><![CDATA[picurl]]></PicUrl>
      <Url><![CDATA[url]]></Url>
    </item>
    <item>
      <Title><![CDATA[title]]></Title>
      <Description><![CDATA[description]]></Description>
      <PicUrl><![CDATA[picurl]]></PicUrl>
      <Url><![CDATA[url]]></Url>
    </item>
  </Articles>
</xml>

```



```
</Articles>
</xml>
```



备注：news 消息中的封面图片既可以是图片文件的链接，也可以是图片的输出流链接。
news 消息一次性最多能够发送 10 条消息，且第一条消息的封面图片将被放大。

(2) 消息体详细说明如表 4.29 所示。

表 4.29 news响应消息格式

参数	说明
ToUserName	成员UserID
FromUserName	企业号CorpID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：news
ArticleCount	图文条数，默认第一条为大图。图文数不能超过10，否则会无响应
Title	图文消息标题
Description	图文消息描述
PicUrl	图片链接，支持JPG、PNG格式，较好的效果为大图360×200，小图200×200（单位：像素）
Url	点击图文消息跳转链接

(3) 示例代码。

利用面向对象的方式，封装 news 响应类 RespNewsMessage.java 和 RespArticle.java，详细代码如下：

```
package myf.caption4.resp;
import java.util.List;
public class RespNewsMessage {

    //接收方账号（收到的 OpenID）
    private String ToUserName;
    //开发者微信号
    private String FromUserName;
    //消息创建时间(整型)
    private long CreateTime;
    //消息类型
    private String MsgType;
    //图文消息个数，限制为 10 条以内
    private int ArticleCount;
    //多条图文消息信息，默认第一条 news 消息为大图
    private List<RespArticle> Articles;
    public String getToUserName() {
        return ToUserName;
    }
    public void setToUserName(String toUserName) {
```

```

        ToUserName = toUserName;
    }
    public String getFromUserName() {
        return FromUserName;
    }
    public void setFromUserName(String fromUserName) {
        FromUserName = fromUserName;
    }
    public long getCreateTime() {
        return CreateTime;
    }
    public void setCreateTime(long createTime) {
        CreateTime = createTime;
    }
    public String getMsgType() {
        return MsgType;
    }
    public void setMsgType(String msgType) {
        MsgType = msgType;
    }
    public int getArticleCount() {
        return ArticleCount;
    }
    public void setArticleCount(int articleCount) {
        ArticleCount = articleCount;
    }
    public List<RespArticle> getArticles() {
        return Articles;
    }
    public void setArticles(List<RespArticle> articles) {
        Articles = articles;
    }
}

```

/**

* 文字类

*/

package myf.caption3.demo3_5;

public class WxArticle {

//图文消息名称

private String title;

//图文消息描述

private String description;

//图片链接, 支持 JPG、PNG 格式, 较好的效果为大图 640×320, 小图 80×80

private String picurl;

//点击图文消息跳转链接

```

private String url;
public String getTitle() {
    return title;
}
public void setTitle(String title) {
    this.title = title;
}
public String getDescription() {
    return description;
}
public void setDescription(String description) {
    this.description = description;
}
public String getPicurl() {
    return picurl;
}
public void setPicurl(String picurl) {
    this.picurl = picurl;
}
public String getUrl() {
    return url;
}
public void setUrl(String url) {
    this.url = url;
}
public WxArticle(String title, String description, String picurl, String
url) {
    super();
    this.title = title;
    this.description = description;
    this.picurl = picurl;
    this.url = url;
}
}

```



备注：消息的加密及 XML，请参照 4.3 节和 4.6 节。

4.10 案例：企业通讯录快速搜索

通过前面的学习，我们已经了解了被动回调模式的消息接收，本节将从实际的业务中学习回调模式的开发——企业通讯录快速搜索，学习“文本”消息和“进入”事件的接收响应、异步消息的推送，以及响应消息的加密处理等。企业通讯录搜索支持姓名的汉字、拼音全拼、电话号码和部门名称等搜索，如图 4.17 所示。

备注：通讯录企业号默认提供，也可以根据自身需要重新开发；快速检索也可用于工单的快速检索功能。

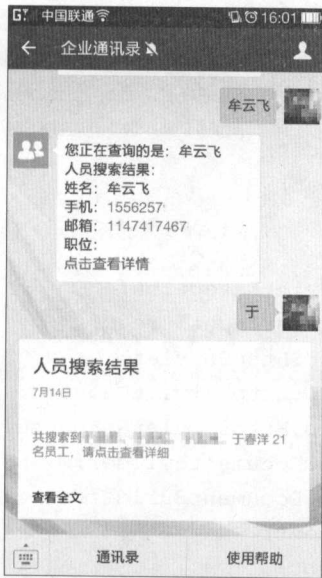


图 4.17 企业通讯录消息被动响应

异步推送能够根据输入的信息，利用异步响应推送多条信息，查询出符合条件的人员信息和部门信息，如图 4.18 所示。

备注：异步响应，在企业号接收消息之后，有时会因为回复时间过长（超过五秒）或需要回复多条消息等原因，先回复空消息，然后再使用主动推送的方式响应员工。

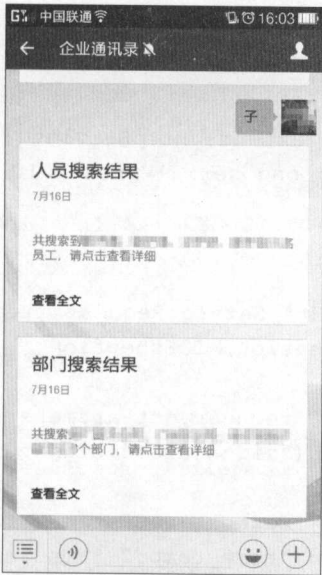


图 4.18 消息响应异步推送

示例代码如下：

```
package myf.caption4.servlet;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.io.StringReader;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import javax.servlet.ServletException;
import javax.servlet.ServletInputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;
import myf.caption4.resp.AddressBook;
import myf.caption4.resp.RespArticle;
import myf.caption4.resp.RespNewsMessage;
import myf.caption4.resp.RespTextMessage;
import myf.caption4.resp.WxParty;
import myf.caption4.QQTool.WXBizMsgCrypt;
/**
 * 回调模式处理类
 *
 */
public class CoreServlet extends HttpServlet {
    private static final long serialVersionUID = 4440739483644821986L;
    /**
     * 请求校验
     */
    public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        //微信加密签名
        String signature = request.getParameter("msg_signature");
        System.out.println("signature:"+signature);
        //时间戳
        String timestamp = request.getParameter("timestamp");
        System.out.println("timestamp:"+timestamp);
        //随机数
        String nonce = request.getParameter("nonce");
```

```

System.out.println("nonce:"+nonce);
//随机字符串
String echostr = request.getParameter("echostr");
System.out.println("echostr:"+echostr);
String sToken = WxUtil.RESP_MESSAGE_TOKEN;
String sCorpID = WxUtil.MESSAGE_CORPID;
String sEncodingAESKey = WxUtil.RESP_MESSAGE_ENCODINGAESKEY;
try {
    WXBizMsgCrypt wxcpt = new WXBizMsgCrypt(sToken, sEncodingAESKey, sCorpID);
    String sEchoStr; //需要返回的明文
    sEchoStr = wxcpt.VerifyURL(signature, timestamp, nonce, echostr);
    System.out.println("verifyurl echostr: " + sEchoStr);
    // 验证 URL 成功, 将 sEchoStr 返回
    PrintWriter out = response.getWriter();
    out.write(sEchoStr);
    out.flush();
    out.close();
} catch (Exception e) {
    //验证 URL 失败, 错误原因请查看异常
    e.printStackTrace();
}
}
/**
 * 处理微信服务器发来的消息
 */
public void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    //读取消息, 执行消息处理
    //使用数据库查询数据, 也可以使用 lucene 查询
    String sReqMsgSig = request.getParameter("msg_signature");
    //时间戳
    String sReqTimeStamp = request.getParameter("timestamp");
    //随机数
    String sReqNonce = request.getParameter("nonce");
    String sToken = WxUtil.RESP_MESSAGE_TOKEN;
    String sCorpID = WxUtil.MESSAGE_CORPID;
    String sEncodingAESKey = WxUtil.RESP_MESSAGE_ENCODINGAESKEY;
    try {
        //POST 请求的密文数据
        //sReqData = HttpUtils.PostData();
        ServletInputStream in = request.getInputStream();
        BufferedReader reader = new BufferedReader(new InputStreamReader(in));
        String sReqData = "";
        String itemStr = ""; //作为输出字符串的临时串, 用于判断是否读取完毕
        while (null != (itemStr = reader.readLine())) {
            sReqData += itemStr;

```

```

    }
    //对消息进行处理获得明文
    WXBizMsgCrypt wxcpt = new WXBizMsgCrypt(sToken, sEncodingAESKey, sCorpID);
    String sMsg = wxcpt.DecryptMsg(sReqMsgSig, sReqTimeStamp, sReqNonce,
sReqData);

    //输出解密后的文件
    //System.out.println("after decrypt msg: " + sMsg);
    //TODO: 解析出明文 XML 标签的内容进行处理
    //For example:
    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
    DocumentBuilder db = dbf.newDocumentBuilder();
    StringReader sr = new StringReader(sMsg);
    InputSource is = new InputSource(sr);
    Document document = db.parse(is);
    Element root = document.getDocumentElement();
    //需要排重, 进行排重
    //有 msgid 的消息推荐使用 msgid 排重
    //事件类型消息推荐使用 FromUserName + CreateTime 排重
    //判断类型
    NodeList nodelist_msgType = root.getElementsByTagName("MsgType");
    String recieveMsgType = nodelist_msgType.item(0).getTextContent();
    String content="";
    String content2="";
    String sEncryptMsg="";//加密的消息
    String sEncryptMsg2="";//加密的消息
    int resultCount=1;//发送消息的数量
    int resultCount2=1;//发送消息的数量
    //如果是文本消息, 则需根据输入的文本消息查询数据
    if("text".equals(recieveMsgType)){
        //获得内容
        NodeList nodelist1 = root.getElementsByTagName("Content");
        String context=(nodelist1.item(0).getTextContent()+"").trim();
        String
context2=(nodelist1.item(0).getTextContent()+"").trim();
        //回复人
        NodeList nodelist_fromUser = root.getElementsByTagName
("FromUserName");
        String mycreate = nodelist_fromUser.item(0).getTextContent();
        //数据库查询人员 List
        List<AddressBook> documentList = service.retrieveByKeys
(context);
        //数据库查询部门 List
        List<WxParty> departMentdocumentList = partyService.retrieve
(null,extql,params,null,null);
        if((null==documentList||0==documentList.size())&&(null==
departMentdocumentList||0==departMentdocumentList.size())){

```

的信息, 请再试试? ";

```

        //时间
        String time=new Date().getTime()+"";
        content=""+content;
        System.out.println(content);
        //生成一个被动响应的消息
        RespTextMessage txtMsg= new RespTextMessage();
        txtMsg.setContent(content);//文字内容
        txtMsg.setCreateTime(Long.valueOf(time));//创建时间
        txtMsg.setFromUserName(sCorpID);//消息来源
        txtMsg.setMsgType(WxUtil.RESP_MESSAGE_TYPE_TEXT);
        txtMsg.setToUserName(mycreate);
        String sRespData=WxUtil.messageToXml(txtMsg);
        sEncryptMsg = wxcpt.EncryptMsg(sRespData, time, sReqNonce);
        //设置响应消息日志
        //保存数据库
    }else if(documentList.size()==1){
        //查询输出结果, 一条数据
        AddressBook doc = documentList.get(0);
        context="您正在查询的是: "+context+" \r\n"
        +"人员搜索结果: \r\n"
        +"姓名: "+doc.getName()+"\r\n"
        +"手机: "+(doc.getMobile()==null?"":doc.getMobile())+
"\r\n"
        +"邮箱: "+(doc.getEmail()==null?"":doc.getEmail())+"\r\n"
        +"职位: "+(doc.getPosition()==null?"":doc.getPosition())+
"\r\n"
        +"<a href='"+WxUtil.webUrl+"/bookDetailSlt? userId=
"+doc.getUserid()+">点击查看详情</a>";
        //设置响应内容
        content = context;
        //时间
        String time=new Date().getTime()+"";
        //生成一个被动响应的消息
        RespTextMessage txtMsg= new RespTextMessage();
        txtMsg.setContent(content);//文字内容
        txtMsg.setCreateTime(Long.valueOf(time));//创建时间
        txtMsg.setFromUserName(sCorpID);//消息来源
        txtMsg.setMsgType(WxUtil.RESP_MESSAGE_TYPE_TEXT);
        txtMsg.setToUserName(mycreate);
        String sRespData=WxUtil.messageToXml(txtMsg);
        sEncryptMsg = wxcpt.EncryptMsg(sRespData, time, sReqNonce);

        //设置响应消息日志
        //保存数据
    }
}

```



```

    }else if(documentList.size()>1){
        //生成文章描述
        String discript="";
        String codeStr="";//搜索到的人员
        for (int i = 0; i < documentList.size(); i++) {
            AddressBook item = documentList.get(i);
            if(0!=i){
                if(i<=3){
                    discript+="、 ";
                }
                codeStr+=",";
            }
            codeStr+=item.getUserid();
            //设置最多显示 4 个人
            if(i<=3){
                discript+=item.getName();
            }
        }
        //设置响应内容
        content = "共搜索到"+discript+" "+documentList.size()+" 名
员工, 请点击查看详细";

        //时间
        String time=new Date().getTime()+"";
        //发送 news 消息
        RespNewsMessage news=new RespNewsMessage();
        news.setFromUserName(WxUtil.MESSAGE_CORPID);//消息来源
        news.setMsgType(WxUtil.RESP_MESSAGE_TYPE_NEWS);
        news.setToUserName(mycreate);//回复人
        news.setCreateTime(Long.valueOf(time));//创建时间
        news.setArticleCount(1);//文章数量
        List<RespArticle> articles = new ArrayList<RespArticle>();
        RespArticle article=new RespArticle();
        article.setTitle("人员搜索结果");
        //查看消息详细的链接
        article.setUrl(WxUtil.webUrl+"/wx/BookSlt?
userStr="+codeStr);

        article.setDescription(content);
        articles.add(article);
        news.setArticles(articles);
        String sRespData=WxUtil.messageToXml(news);
        //System.out.println(sRespData);
        sEncryptMsg = wxcpt.EncryptMsg(sRespData, time, sReqNonce);
        //设置响应消息日志
        //保存数据库
    }

```

```

//推送部门信息
if (null!=departMentdocumentList&&departMentdocumentList.
size()==1){
    //查询输出结果，一条数据
    WxParty doc =departMentdocumentList.get(0);
    context2="您正在查询的是："+context2+"\r\n"
    +"部门搜索结果：\r\n"
    +"部门："+doc.getPartyName()+"\r\n"
    +"电话："+(doc.getTelno()==null?"":doc.getTelno()+"\r\n"
    +"负责人："+(doc.getManager()==null?"":doc.getManager()+
"\r\n"
    +"联系地址："+(doc.getAddress()==null?"":doc.getAddress()+
"\r\n"
    +"<a href='"+WxUtil.webUrl+"/wx/BookSlt?
        userId="+doc.getPartyId()+"'>点击查看详情</a>";
    //设置响应内容
    content2 = context2;
    //时间
    String time=new Date().getTime()+"";
    //生成一个被动响应的消息
    RespTextMessage txtMsg= new RespTextMessage();
    txtMsg.setContent(content2);//文字内容
    txtMsg.setCreateTime(Long.valueOf(time));//创建时间
    txtMsg.setFromUserName(sCorpID);//消息来源
    txtMsg.setMsgType(WxUtil.RESP_MESSAGE_TYPE_TEXT);
    txtMsg.setToUserName(mycreate);
    String sRespData=WxUtil.messageToXml(txtMsg);
    sEncryptMsg2=wxcpt.EncryptMsg(sRespData, time, sReqNonce);
    //设置响应消息日志，保存数据
}else if (null!=departMentdocumentList&&departMentdocumentList.
size())>1){
    //消息数量决定消息反馈的类型
    resultCount2=departMentdocumentList.size();
    //生成文章描述
    String discript="";
    String codeStr="";//搜索到的人员
    for (int i = 0; i < departMentdocumentList.size(); i++) {
        WxParty item = departMentdocumentList.get(i);
        if(0!=i){
            if(i<=2){
                discript+="、";
            }
            codeStr+=", ";
        }
        codeStr+=item.getPartyId();
    }
    //设置最多显示4个人

```

```

        if(i<=2){
            discript+=item.getPartyName();
        }
    }
    //设置响应内容
    content2 = "共搜索到"+discript+" "+departMentdocumentList.
size()+"个部门, 请点击查看详细";
    //时间
    String time=new Date().getTime()+"";
    //发送 news 消息
    RespNewsMessage news=new RespNewsMessage();
    news.setFromUserName(WxUtil.MESSAGE_CORPID);//消息来源
    news.setMsgType(WxUtil.RESP_MESSAGE_TYPE_NEWS);
    news.setToUserName(mycreate);//回复人
    news.setCreateTime(Long.valueOf(time));//创建时间
    news.setArticleCount(1);//文章数量
    List<RespArticle> articles = new ArrayList<RespArticle>();
    RespArticle article=new RespArticle();
    article.setTitle("部门搜索结果");
    article.setUrl(WxUtil.webUrl+"/wx/BookSlt? userStr="+
codeStr);

    article.setDescription(content2);
    articles.add(article);
    news.setArticles(articles);
    String sRespData=WxUtil.messageToXml(news);
    //System.out.println(sRespData);
    sEncryptMsg2 = wxcpt.EncryptMsg(sRespData, time,
sReqNonce);

    //设置响应消息日志, 保存数据
}

//可以在这里使用线程保存数据库
//因为响应必须在五秒钟内进行

//!!!!!!!!!!!!!!!!!!!!!!设置回复!!!!!!!!!!!!!!
//-----被动响应只能响应一条, 另一条需要主动推送-----
if(null!=content2&&"".equals(content2)&&>null!=content&&"".
equals(content)){
    //被动响应一条
    PrintWriter out = response.getWriter();
    out.write(sEncryptMsg);
    out.flush();
    out.close();
    //主动推送一条, 异步响应, 主动推送
    if(resultCount2<=1){
        //如果返回结果是一条数据, 主动推送文本消息

```



```

        WxTextMessage text = new WxTextMessage();
        text.setContent(content2);
        text.setMsgtype(WxUtil.REQ_MESSAGE_TYPE_TEXT);
        text.setTouser(respEvent2.getToUser());
        text.setAgentid(WxUtil.AGENT_ID_ADDRESSBOOK);
        text.setSafe("0");
        WxUtil util=new WxUtil();
        util.sendReqMsg(text);
    }else{
        //如果返回结果是多条数据, 主动推送 news 消息
        List<WxArticle> newsList= new ArrayList<WxArticle>();
        WxArticle article=new WxArticle();
        article.setTitle(respMessage2.getTitle());
        article.setUrl(respMessage2.getUrl());
        article.setDescription(respMessage2.getContent());
        newsList.add(article);
        WxNewsMessage news = new WxNewsMessage();
        news.setMsgtype(WxUtil.REQ_MESSAGE_TYPE_NEWS);
        news.setTouser(respEvent2.getToUser());
        news.setAgentid(WxUtil.AGENT_ID_ADDRESSBOOK);
        news.setNews(newsList);
        WxUtil util=new WxUtil();
        util.sendReqMsg(news);
    }
    }else if(null!=content&&"".equals(content)&&(null==content2||"".equals(content2))){
        //输出
        PrintWriter out = response.getWriter();
        out.write(sEncryptMsg);
        out.flush();
        out.close();
    }else if(null!=content2&&"".equals(content2)&&(null==content||"".equals(content))){
        //输出
        PrintWriter out = response.getWriter();
        out.write(sEncryptMsg2);
        out.flush();
        out.close();
    }
}

}else if("event".equals(recieveMsgType)){//如果是时间消息
    //获得事件类型
    NodeList nodelist1 = root.getElementsByTagName("Event");
    String event_type = nodelist1.item(0).getTextContent();
    if("click".equals(event_type)){
        //获得 event_key

```



```

        NodeList node_EventKey = root.getElementsByTagName
("EventKey");
        String eventKey = node_EventKey.item(0).getTextContent();
        if ("ADDRESSBOOK_HELP".equals(eventKey)) {
            //响应消息, 发送 news 消息
            RespNewsMessage news = new RespNewsMessage();
            news.setFromUserName(WxUtil.MESSAGE_CORPID);
            news.setMsgType(WxUtil.RESP_MESSAGE_TYPE_NEWS);
            NodeList enter_people_note = root.getElementsByTagName
("FromUserName");
            String enter_people = enter_people_note.item(0).
getTextContent();
            news.setToUserName(enter_people); //回复人
            long time = new Date().getTime();
            news.setCreateTime(time); //创建时间
            news.setArticleCount(1); //文章数量
            List<RespArticle> articles = new ArrayList<RespArticle>();
            RespArticle article = new RespArticle();
            article.setTitle("企业通讯录使用须知");
            article.setPicUrl(WxUtil.webUrl + "/addbotb.png");
            article.setUrl(WxUtil.webUrl + "/bookExplain.jsp");
            article.setDescription("通讯录随时随地按组织架构展示, 清晰
明了, 可自定义拼音、汉字、手机号码等搜索");

            articles.add(article);
            news.setArticles(articles);
            String sRespData = WxUtil.messageToXml(news);
            //System.out.println(sRespData);

            content = sEncryptMsg = wxcpt.EncryptMsg(sRespData,
time + "", sReqNonce);
        }
    }
    //!!!!!!!!!!!!!!!!!!!!!!设置回复!!!!!!!!!!!!!!
    //-----
    if (null != content && "".equals(content)) {
        //输出
        PrintWriter out = response.getWriter();
        out.write(sEncryptMsg);
        out.flush();
        out.close();
    }
}

} catch (Exception e) {
    //TODO

```

```
//解密失败，失败原因请查看异常
e.printStackTrace();
}
```

备注：本案例结合了“4.7.2 接收 text 消息”、“4.8.4 接收进入应用事件”、“4.9.2 被动响应 text 消息”、“4.9.6 被动响应 news 消息”、主动推送“3.5.2 Text 消息”以及主动推送“3.5.7 News 消息”。由于本案例结合较多，如果对于部分知识较为模糊，则建议到相关章节查看详细内容，这里将不再说明。

第5章

JSAPI 模式

JSAPI 模式使应用功能更加丰富。前面我们学习了微信企业号与企业员工之间的消息传递方式，本章将重点学习 JSAPI 模式下如何丰富企业号的应用功能，增强客户应用的可定制化，学习微信内置浏览器的调用以及微信 JS-SDK 各接口的使用。

本章涉及的主要知识点有：

- JSAPI 模式介绍：学会 JSAPI 模式基础知识。
- 接口接入及权限验证：学会如何正确通过权限验证，引入微信 JS 接口。
- JS 接口调用：学会使用常用的 JS 接口。
- 接口列表说明：了解微信 JSAPI 模式下具有的接口功能。
- 业务与接口的实际应用：通过本章最后的示例，学习微信 JS 接口的使用方法；通过本章所学的知识，正确使用微信 JS 接口。

5.1 JSAPI 模式介绍

在介绍 JSAPI 模式之前，首先需要介绍一下微信内置浏览器。可能很多人注意到了，在打开微信“朋友圈”链接的时候会出现进度条，如图 5.1 所示，这实际上就是微信内置浏览器访问页面的进度。也就是说，“朋友圈”是通过微信内置的浏览器访问的手机页面，并且微信浏览器是在微信安装时内置在微信中的。

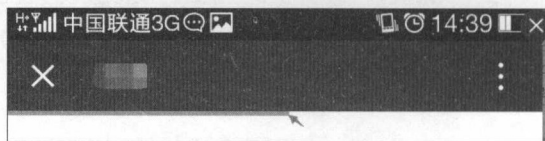


图 5.1 微信内置浏览器进度条



注意：iPhone（苹果）和 Android（安卓）的微信内置浏览器不同，安卓手机上的微信使用的是 QQ 浏览器 X5 内核，苹果手机上的微信使用的则是 Safari 浏览器。

JSAPI 模式是通过调用微信 JS-SDK 开发手机 Web 页面的模式，本质上亦是开发 B/S

(Browser/Server, 浏览器/服务器模式) 服务, 只是业务上较以往的 PC 业务更加方便, 功能上也稍具差异。在微信 JSAPI 模式下, 不仅可以调用微信拍照、选图、语音、位置等手机功能, 还可以实现微信分享、扫一扫等微信特有的功能, 同时, 可以使用 HTML5 完成页面效果的丰富, 实现更加完美的用户体验。

5.2 页面接口引入

在了解了 JSAPI 模式后, 本节将介绍如何接入微信 JS-SDK, 以及学习 config 接口权限配置和结果事件的处理。

微信 JS-SDK 接入准备:

- 通过微信管理端完成“可信域名”的配置。

微信 JS-SDK 接入流程:

- JSP 页面中引入 jweixin-1.1.0.js 文件;
- 通过 config 接口注入权限验证配置信息;
- 微信处理验证信息, 返回处理结果;
- 验证成功调用 ready 接口, 继续自定义处理;
- 验证失败调用 error 接口, 继续自定义处理。

5.2.1 配置“可信域名”

所有的 JS 接口只能在企业号应用的“可信域名”下调用(包括子域名), 所以域名的配置是必需的。配置方法: 打开微信管理端, 单击“应用中心”, 选择当前开发的应用, 单击进入应用详细页面, 在“可信域名”中填写, 如图 5.2 所示。服务号中可以填写三个可信域名, 而企业号中需要对每个应用进行配置, 且仅能配置一个可信域名。

可信域名

设置应用可信域名后, 应用可使用以下功能:

1. 应用页面使用微信JS-SDK查看接入文档
2. 可作为应用OAuth2.0网页授权功能的回调域名查看接入文档
3. 用户在该域名上进行输入时, 不会出现风险安全提示框(仅对认证号有效)

域名要求:

1. 设置的应用域名须通过ICP备案的验证
2. 请使用二级或二级以上域名, 如weixin.qq.com

填写应用的域名地址, 如: qy.weixin.qq.com:8080

图 5.2 配置“可信域名”

注意: 可信域名必须使用 ICP (电信与信息服务业务经营许可证) 备案的域名。如果访问的外网域名含有非 80 端口号, 则需要在可信域名地址中添加端口号。

5.2.2 引入微信 JS 文件

完成域名配置后，我们就进入正式的页面开发，首先需要引入微信的 JS 文件，这里我们可以在线引入 jweixin-1.1.0.js，示例代码如下：

```
<script type="text/javascript" src="http://res.wx.qq.com/open/js/jweixin-1.0.0.js"></script>
```



备注：支持使用 AMD/CMD 标准模块加载方法加载。

也可以将 JS 文件下载到自己的工程中进行引入：

```
<script type="text/javascript" src="<%=request.getContextPath()%>/wx/js/jweixin-1.0.0.js">
</script>
```

JS 文件的下载：可以在 IE 浏览器中直接输入 <http://res.wx.qq.com/open/js/jweixin-1.1.0.js>，即可下载链接。



使用技巧：request 为 JSP 九大内置对象之一，<%%>为 JSP 页面动态 Java 代码执行标识框，request.getContextPath()获取当前系统路径，防止因路径变更造成的代码返工。

5.2.3 权限验证

微信 JS-SDK 权限验证主要分为三步：

01 获取调用票据 jsapi_ticket。

02 生成 JS-SDK 权限验证签名。

03 页面 config 接口配置注入。

1. 获取调用票据 jsapi_ticket

jsapi_ticket 是企业号调用微信 JS 接口的临时票据，目前有效期为 7200 秒（以返回结果中的 expires_in 为准），细心的读者可能已经发现了，这与第 3 章中的 Token 有效时间相同。不错，jsapi_ticket 的有效时间与 Token 的有效时间相同，并且都具有调用频率限制，所以需要缓存处理，处理办法与 Token 处理方法一致：采用全局变量的方式。

```
//JSAPI 模式：全局变量 jsapi_ticket
public static String jsapi_ticket;
// JSAPI 模式：请求 jsapi_ticket 的时间
public static Date jsapi_ticket_date;
```

jsapi_ticket 的获取大致分为两步：

01 获取 Token；

02 利用 Token 获取 jsapi_ticket，并进行全局静态变量保存。

（1）获取全局缓存变量 Token。

Token 的获取方式在第 3 章“Access Token 的缓存处理”做过讲解，这里我们直接进行调用。

```
String token=getTokenFromWx();//获得 Token
```

(2) 获取、缓存处理 jsapi_ticket。

jsapi_ticket 的请求采用 HTTP GET 的方式，请求地址为：https://qyapi.weixin.qq.com/cgi-bin/get_jsapi_ticket?access_token=ACCESS_TOKEN

调用成功后，返回 JSON 数据，数据格式如下：

```
{
    "errcode":0,
    "errmsg":"ok",
    "ticket":"bXLdikRXVbTPdHSM05e5u5sUoXNKd8-4lZO3MhKoyN5OfkWITDGgnr2fwJ0m9E8NYz
WKVZvdVtaUgWvsdshFKA",
    "expires_in":7200
}
```

返回的 JSON 数据结果参数说明如表 5.1 所示。

表 5.1 Array类常用属性/方法说明

属性/方法	说明
errcode	返回码，0表示成功
errmsg	返回码的文本描述信息
ticket	微信JS接口调用的临时票据
expires	临时票据的有效期，单位为秒

jsapi_ticket 获取及缓存处理源码如下：

```
/**
 * 从微信获得 jsapi_ticket
 * @return 返回 ticket
 */
public String getJsapiTicketFromWx(){
    String token=getTokenFromWx();//token
    //1. 判断 jsapi_ticket 是否存在，不存在则直接申请
    //2. 判断时间是否过期，过期（>=7200 秒）申请，否则不用请求直接返回以后的 Token
    if(null==jsapi_ticket||"".equals(jsapi_ticket)||(new
Date().getTime()-jsapi_ticket_date.getTime())>=(7000*1000)){

        CloseableHttpClient httpclient = HttpClients.createDefault();
        try {
            //利用 GET 形式获得 Token
            HttpGet httpget = new HttpGet("https://qyapi.weixin.qq.com/
cgi-bin/get_jsapi_ticket?access_token="+token);
            // Create a custom response handler
            ResponseHandler<JSONObject> responseHandler = new ResponseHandler
<JSONObject>() {

                public JSONObject handleResponse(
```

```

        final HttpResponse response) throws ClientProtocolException,
IOException {
    int status = response.getStatusLine().getStatusCode();
    if (status >= 200 && status < 300) {
        HttpEntity entity = response.getEntity();
        if (null!=entity){
            String result= EntityUtils.toString(entity);
            //根据字符串生成 JSON 对象
            JSONObject resultObj = JSONObject.fromObject(result);
            return resultObj;
        }else{
            return null;
        }
    } else {
        throw new ClientProtocolException("Unexpected response
status: " + status);
    }
}

};
//返回的 JSON 对象
JSONObject responseBody = httpclient.execute(httpget, responseHandler);
if (null!=responseBody) {
    jsapi_ticket= (String) responseBody.get("ticket");//返回 Token
}
jsapi_ticket_date=new Date();
httpclient.close();
} catch (Exception e) {
    e.printStackTrace();
}
}
return jsapi_tick
}

```



说明：有效时间为 7200 秒，这里设置了大于等于 7000 秒，目的是为了防止服务器事件延迟导致的服务异常，预留一定时间，方便服务器维护人员及时处理。

2. 生成 JS-SDK 权限验证签名

在获取到 jsapi_ticket 之后，我们就利用有效的 jsapi_ticket、noncestr(随机字符串)、timestamp(时间戳)和 URL(当前网页的 URL，不包含#及其后面部分)四个变量完成权限验证签名，签名的规则如下：

将参与签名的字段 noncestr、有效的 jsapi_ticket、timestamp 和 URL 按照字段名的 ASCII 码从小到大排序(字典序)后，使用 URL 键值对的格式(即 key1=value1&key2=value2...)拼接成字符串 string1。这里需要注意的是，所有参数名均为小写字母。对 string1 做 sha1 加密，字段名和字段值均采用原始值，不进行 URL 转义。

注意：生成验证签名必须在后台进行，使用 Angular.js 的读者也需要使用 \$http 进行后台签名生成。

权限验证签名示例代码如下：

```

/****微信 js 签名*****/
public static Map<String, String> sign(String jsapi_ticket, String url) {
    Map<String, String> ret = new HashMap<String, String>();
    String nonce_str = create_nonce_str();
    String timestamp = create_timestamp();
    String string1;
    String signature = "";
    //注意，这里参数名必须全部小写，且必须有序
    string1 = "jsapi_ticket=" + jsapi_ticket +
        "&noncestr=" + nonce_str +
        "&timestamp=" + timestamp +
        "&url=" + url;

    try
    {
        MessageDigest crypt = MessageDigest.getInstance("SHA-1");
        crypt.reset();
        crypt.update(string1.getBytes("UTF-8"));
        signature = byteToHex(crypt.digest());
    }
    catch (NoSuchAlgorithmException e)
    {
        e.printStackTrace();
    }
    catch (UnsupportedEncodingException e)
    {
        e.printStackTrace();
    }
    ret.put("url", url);
    ret.put("jsapi_ticket", jsapi_ticket);
    ret.put("nonceStr", nonce_str);
    ret.put("timestamp", timestamp);
    ret.put("signature", signature);
    return ret;
}

private static String byteToHex(final byte[] hash) {
    Formatter formatter = new Formatter();
    for (byte b : hash)
    {
        formatter.format("%02x", b);
    }
    String result = formatter.toString();
}

```



```
        formatter.close();
        return result;
    }
    private static String create_nonce_str() {
        return UUID.randomUUID().toString();
    }
    private static String create_timestamp() {
        return Long.toString(System.currentTimeMillis() / 1000);
    }
}
```

算法调用示例代码如下:


```
//生成微信 JS 授权
String jsapi_ticket=msgUtil.getJsapiTicketFromWx();//签名
//获得当前页面请求的 URL
HttpServletRequest req = ServletActionContext.getRequest();
String url = WxUtil.webUrl //域名地址, 自己的外网请求的域名, 如:
http://www.baidu.com/Demo
        + req.getServletPath() //请求页面或其他地址
        + "?" + (req.getQueryString()); //参数
//输出请求地址
System.out.println(url);
//获得 jsapi_ticket
Map<String, String> ret = WxUtil.sign(jsapi_ticket, url);
//放入页面信息
req.setAttribute("corpId", WxUtil.RESP_MESSAGE_CORPID);//企业号 CorpId
req.setAttribute("str1", ret.get("signature"));//权限验证签名
req.setAttribute("time", ret.get("timestamp"));//时间戳
req.setAttribute("nonceStr", ret.get("nonceStr"));//随机串
```

注意: 当前页面请求的 URL 需要动态获取, 不要手动写成固定变量。固定变量与第 8 章介绍的 OAuth 2.0 身份验证结合后, 会造成权限签名失败, 同时, 固定变量也无法应用于 URL 变化的应用中, URL 的动态变化将导致授权签名失败。

3. 页面 config 接口配置注入

config 是一个客户端异步操作接口, 请求页面将通过 config 接口注入权限配置信息, 只需将如下代码写入<script></script>标签中即可:

```
wx.config({
    debug: false, //关闭调试模式
    appId: '${corpId}', //必填, 企业号的唯一标识, 此处填写企业号 CorpID
    timestamp: "${time}", //必填, 生成签名的时间戳
    nonceStr: '${nonceStr}', //必填, 生成签名的随机串
    signature: '${str1}', //必填, 签名, 见附录 A
    jsApiList: ['hideOptionsMenu'] //必填, 需要使用的 JS 接口列表
});
```

 说明: `${time}` 为 EL 表达式, 意思是获得 `req.setAttribute("time", ret.get("timestamp"))` 中设置的 `time` 数据

5.2.4 验证成功事件

权限验证成功之后, 将执行 `ready` 接口, 所有接口调用都必须在 `config` 接口获得结果之后。`config` 是一个异步操作, 所以对于需要在页面加载时就调用的接口, 需要把相关接口放在 `ready` 函数中调用。如果是用户触发时才调用的接口, 则可以直接调用, 不需要放入 `ready` 中执行。例如示例“进入页面后立即隐藏右上角菜单按钮”:

```
wx.ready(function() {
    //隐藏右上角菜单
    wx.hideOptionMenu();
});
```

5.2.5 验证失败事件

`config` 信息验证失败会执行 `error` 函数。例如, 签名过期导致验证失败, 具体错误信息可以打开 `config` 的 `debug` 模式中查看, 也可以在返回的 `res` 参数中查看, 对于 SPA 可以在这里更新签名:

```
wx.error(function(res) {
    // 处理失败消息
});
```

5.3 Debug 调试及基础接口说明

学习完接口引入之后, 对于开发来说, 比较重要的一项是断点调试, 调试接口返回信息, 调试接口是否可以正常调用, 本节我们将学习如何在 JSAPI 模式下进行 Debug 调试以及基础接口调用。

5.3.1 Debug 调试模式开启

Debug 模式与 Java、C# 等开发一样, 默认调试模式是关闭, 表现形式为权限验证页面中的“`debug: false`”, 开启方式如下所示:

```
wx.config({
    debug: true, //开启调试模式
    appId: '${corpId}', //必填, 企业号的唯一标识, 此处填写企业号 CorpID
    timestamp: "${time}", //必填, 生成签名的时间戳
    nonceStr: '${nonceStr}', //必填, 生成签名的随机串
    signature: '${str1}', //必填, 签名, 见附录 A
    jsApiList: ['hideOptionMenu'] //必填, 需要使用的 JS 接口列表
});
```

将 debug 设置为 true 即可，这样调用所有 APL 的返回值会在客户端弹出消息提示，输出接口信息，如图 5.3 所示。若要查看传入的参数，可以在 PC 端打开，参数信息会通过 log 打印出来，仅在 PC 端时才能打印。

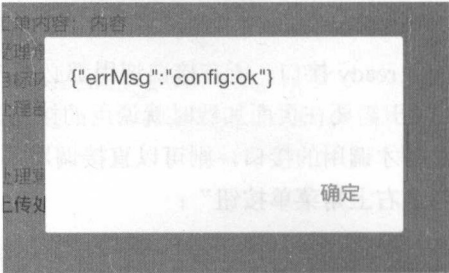


图 5.3 微信调试模式下接口返回值显示

备注：jsApiList 中的多个函数用逗号分隔。

5.3.2 判断当前客户端版本是否支持指定 JS 接口

随着微信客户端的升级以及接口的开放，必然存在旧客户端无法兼容新客户端接口的问题，从而导致接口无法使用。这类问题可能使开发者误以为自己的调用出现问题，进而耗费太多时间。这里介绍一个基础接口——微信 JS 检查接口，接口示例如下：

```
wx.config({
  debug: true, //开启调试模式
  appId: '${corpid}', //必填，企业号的唯一标识，此处填写企业号 CorpID
  timestamp: '${time}', //必填，生成签名的时间戳
  nonceStr: '${nonceStr}', //必填，生成签名的随机串
  signature: '${str1}', //必填，签名，见附录 A
  jsApiList: ['chooseImage', 'checkJsApi'] //必填，需要使用的 JS 接口列表
});
wx.ready(function(){
  wx.checkJsApi({
    jsApiList: ['chooseImage'], //需要检测的 JS 接口列表
    success: function(res) {
      //调试模式下，将直接弹出消息提示，输出接口返回信息
    }
  });
});
```

返回结果将以键值对的形式返回，可用的 API 值为 true，不可用的 API 值为 false。例如，{"checkResult":{"chooseImage":true},"errMsg":"checkJsApi:ok"}，实际显示结果如图 5.4 所示。

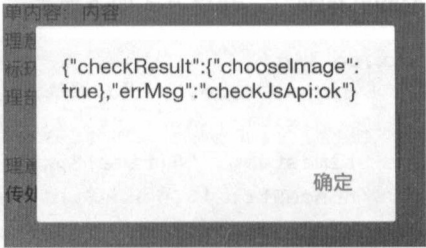


图 5.4 微信调试模式下接口返回值显示

5.3.3 接口通用函数

微信 JS-SDK 的每个接口除了前几节常用到的 ready、error 函数之外，还有 fail、complete、cancel、trigger 函数，详细说明如下：

- ❑ success: 接口调用成功时执行的回调函数。
- ❑ fail: 接口调用失败时执行的回调函数。
- ❑ complete: 接口调用完成时执行的回调函数，无论成功或失败都会执行。
- ❑ cancel: 用户点击取消时的回调函数，仅部分用户有取消操作的 API 才会用到。
- ❑ trigger: 监听 Menu 中的按钮点击时触发的方法，该方法仅支持 Menu 中的相关接口。

以上几个函数都带有一个参数，类型为对象，其中除了每个接口本身返回的数据之外，还有一个通用属性 errMsg，其值格式如下：

- ❑ 调用成功时: "xxx:ok"，其中 xxx 为调用的接口名。
- ❑ 用户取消时: "xxx:cancel"，其中 xxx 为调用的接口名。
- ❑ 调用失败时: 其值为具体错误信息。

! 注意：不要尝试在 trigger 中使用 Ajax 异步请求修改本次分享的内容，因为客户端分享操作是一个同步操作，这时候使用 Ajax 的回包会还没有返回。

5.4 微信 JS-SDK 接口说明

在使用微信 JS-SDK 对应的 JS 接口前，需要根据账号类型（分为注册号和认证号）查看是否已获得对应的 JS 接口权限，其中认证号拥有更多的 JS-SDK 权限，具体权限及接口详细信息如表 5.2 所示。

表 5.2 微信JS-SDK权限接口说明

功能	接口	接口函数	认证号	注册号
基础接口	判断当前客户端版本是否支持指定JS接口	checkJsApi	有	有
企业号	创建企业会话	openEnterpriseChat	有	有
	打开企业通讯录选人	openEnterpriseContact	有	有
分享接口	获取“分享到朋友圈”按钮点击状态及设置分享内容接口	onMenuShareTimeline	有	无
	获取“分享给朋友”按钮点击状态及设置分享内容接口	onMenuShareAppMessage	有	无
	获取“分享到QQ”按钮点击状态及设置分享内容接口	onMenuShareQQ	有	无
	获取“分享到腾讯微博”按钮点击状态及设置分享内容接	onMenuShareWeibo	有	无

续表

功能	接口	接口函数	认证号	注册号
图像接口	本地选图或拍照接口	chooseImage	有	有
	图片预览接口	previewImage	有	有
	上传图片接口	uploadImage	有	有
	下载图片接口	downloadImage	有	有
音频接口	开始录音接口	startRecord	有	有
	停止录音接口	stopRecord	有	有
	监听录音自动停止接口	onVoiceRecordEnd	有	有
	播放录音接口	playVoice	有	有
	暂停播放接口	pauseVoice	有	有
	停止播放接口	stopVoice	有	有
	监听语音播放完毕接口	onVoicePlayEnd	有	有
	上传语音接口	uploadVoice	有	有
	下载语音接口	downloadVoice	有	有
智能接口	识别音频并返回识别结果接口	translateVoice	有	有
设备信息	获取网络状态接口	getNetworkType	有	有
地理位置	查看地理位置地图接口	openLocation	有	有
	获取地理位置接口	getLocation	有	有
界面操作	隐藏右上角菜单接口	hideOptionsMenu	有	有
	显示右上角菜单接口	showOptionsMenu	有	有
	关闭当前窗口接口	closeWindow	有	有
	批量隐藏菜单项接口	hideMenuItems	有	有
	批量显示菜单项接口	showMenuItems	有	有
	隐藏所有非基本菜单项接口	hideAllNonBaseMenuItem	有	有
	显示所有被隐藏的非基本菜单项接口	showAllNonBaseMenuItem	有	有
微信扫一扫	扫一扫接口	scanQRCode	有	有



注意：请不要有诱导分享等违规行为，对于诱导分享行为微信将永久回收企业号接口权限或其他更严厉制裁。

5.5 权限接口应用

前面我们已经学习了接口引入，了解了 JSAPI 模式下的 JS-SDK 接口情况，接下来我们将深入了解一些常用接口，学习如何调用常用的接口。

5.5.1 隐藏右上角菜单

企业号服务于企业、政府、组织等内部人员，因而业务场景中会出现禁止分享的需求。首先需要隐藏右上角的分享功能，防止信息的泄露。当然这只是一步简单的防止信息泄露的措施，在后面我们还会介绍一些其他的数据安全措施，隐藏菜单代码如下所示：

```
wx.config({
  debug: false, //开启调试模式,调用的所有 API 的返回值会在客户端弹出消息提示,输出
//接口信息,若要查看传入的参数,可以在 PC 端打开,参数信息会通过 log 打印出来,仅在 PC 端时
//才能打印
  appId: '${CorpId}', //必填,企业号的唯一标识,此处填写企业号 CorpID
  timestamp: '${time}', //必填,生成签名的时间戳
  nonceStr: '${nonceStr}', //必填,生成签名的随机串
  signature: '${str1}', //必填,签名,见附录 A
  jsApiList: ['hideOptionsMenu'] //必填,需要使用的 JS 接口列表,所有 JS 接口列表
//见附录 B
});
wx.ready(function(){
  //config 信息验证后会执行 ready 方法,所有接口调用都必须在 config 接口获得结果之
//后.config 是一个客户端的异步操作,如果需要在页面加载时就调用相关接口,则必须把相关接口
//放在 ready 函数中调用出来确保正确执行.对于用户触发时才调用的接口,则可以直接调用,无须
//放在 ready 函数中.
  //隐藏右上角菜单
  wx.hideOptionsMenu();
  //  //显示右上角菜单
  //  wx.showOptionsMenu();
  //  //关闭当前网页窗口接口
  //  wx.closeWindow();
});
```

对于有个性化显示或者隐藏的客户,可以使用 hideMenuItems、showMenuItems 方法进行批量隐藏、显示功能按钮,示例代码如下:

```
wx.hideMenuItems({
  menuList: [] //要隐藏的菜单项,所有 menu 项见表 5.3
});
wx.showMenuItems({
  menuList: [] //要显示的菜单项,所有 menu 项见表 5.3
});
```

表 5.3 所有菜单项列表说明

类型	功能	菜单项
基本类	举报	menuItem:exposeArticle
	调整字体	menuItem:setFont
	日间模式	menuItem:dayMode
	夜间模式	menuItem:nightMode
	刷新	menuItem:refresh
	查看企业号 (已添加)	menuItem:profile
	查看企业号 (未添加)	menuItem:addContact
传播类	发送给朋友	menuItem:share:appMessage
	分享到朋友圈	menuItem:share:timeline
	分享到QQ	menuItem:share:qq

续表

类型	功能	菜单项
传播类	分享到QQ空间	menuItem:share:QZone
	分享到Weibo	menuItem:share:weiboApp
	收藏	menuItem:favorite
	分享到facebook	menuItem:share:facebook
	分享到QQ空间	menuItem:share:QZone
保护类	调试	menuItem:jsDebug
	编辑标签	menuItem:editTag
	删除	menuItem:delete
	复制链接	menuItem:copyUrl
	原网页	menuItem:originPage
	阅读模式	menuItem:readMode
	在QQ浏览器中打开	menuItem:openWithQQBrowser
	在Safari中打开	menuItem:openWithSafari
	邮件	menuItem:share:email
	一些特殊企业号	menuItem:share:brand

5.5.2 GPS 定位获取位置信息

getLocation 接口通过 GPS 定位获取当前微信人员的位置信息，可以用于微信考勤（第 11 章将详细介绍）、绘制工作轨迹等场景。在早期的开发中，GPS 定位接口不存在 type 属性，type 属性默认为 wgs84 的 GPS 坐标，如果需要火星坐标可传入“gcj02”，方便读者与其他地图系统进行对接：

```
wx.getLocation({
  type: 'wgs84', //
  success: function (res) {
    var latitude = res.latitude; // 纬度，浮点数，范围为 90 ~ -90
    var longitude = res.longitude ; // 经度，浮点数，范围为 180 ~ -180
    var speed = res.speed; // 速度，以米/每秒计
    var accuracy = res.accuracy; // 位置精度
  }
});
```

注意：不同坐标的地图，需要进行坐标转换，否则将出现较大的位置偏移。

微信推出了 openLocation 接口可以直接显示位置信息，这里需要注意的是，openLocation 接口使用的是火星坐标，所以在获取地理位置的时候需要将 type 属性设置为“gcj02”，openLocation 接口调用示例代码如下：

```
wx.openLocation({
  latitude: 0, //纬度，浮点数，范围为 90 ~ -90
  longitude: 0, //经度，浮点数，范围为 180 ~ -180。
```

```

name: '', //位置名
address: '', //地址详情说明
scale: 1, //地图缩放级别, 整形值, 范围从 1~28。默认为最大
infoUrl: '' //在查看位置界面底部显示的超链接, 可点击跳转
});

```

5.5.3 图片处理接口


目前图片上传之后的有效时间为 3 天, 所以在开发系统时需要及时下载图像到自己的文件服务器中, 以防止因逾期而造成资料丢失。文件下载接口的频率限制为 10000 次/天, 注意要合理地使用接口。如果业务需要提高频率限制, 则可以发送邮件到邮箱 weixin-open@qq.com, 主题为【申请多媒体接口调用量】, 同时需要对项目进行简单描述, 并附上产品体验链接, 以及用户量和使用量说明。下面我们一起来看看图像接口的使用:

01 拍照或从手机相册中选图接口:

```

wx.chooseImage({
  count: 1, // 默认 9
  sizeType: ['original', 'compressed'], //可以指定是原图还是压缩图, 默认二者都有
  sourceType: ['album', 'camera'], //可以指定来源是相册还是相机, 默认二者都有
  success: function (res) {
    var localIds = res.localIds; //返回选定照片的本地 ID 列表, localId 可以作
    为 img 标签的 src 属性显示图片
  }
});

```


 **备注:** 这里的 localIds 为数组, 需要 localIds[i] 得到单个本地 ID。

02 上传图片接口:

```

wx.uploadImage({
  localId: '', //需要上传的图片的本地 ID, 由 chooseImage 接口获得
  isShowProgressTips: 1//默认为 1, 显示进度提示
  success: function (res) {
    var serverId = res.serverId; //返回图片的服务器端 ID
  }
});

```

 **备注:** 这里的 serverId 即为微信服务器上的 media_id。

03 下载图片接口:

```

wx.downloadImage({
  serverId: '', //需要下载的图片的服务器端 ID, 由 uploadImage 接口获得
  isShowProgressTips: 1//默认为 1, 显示进度提示
  success: function (res) {

```



```
        var localId = res.localId; //返回图片下载后的本地 ID
    }
});
```



使用技巧：因为微信服务器的时间为 3 天，因此建议使用自己的服务器上的图片。对于上传图片的预览，也不建议使用微信服务图片，建议直接使用手机本地图片。

5.5.4 语音及智能接口

在类型上，语音接口与图像处理接口一样，都属于多媒体文件，所以上传的语音文件在微信服务器上的保存时间也是 3 天，因而重要的客户录音文件需要及时下载保存。接口调用频率也受限于多媒体下载频率，即每天不能超过 10000 次，如果需要调高频率，则必须发送邮件至 weixin-open@qq.com 申请。语音及智能接口操作如下：

01 开始录音接口：

```
wx.startRecord();
```

02 停止录音接口：

```
wx.stopRecord({
    success: function (res) {
        var localId = res.localId;
    }
});
```

03 监听录音自动停止接口：

```
wx.onVoiceRecordEnd({
    complete: function (res) {
        var localId = res.localId;
    }
});
```



注意：录音时间超过一分钟没有停止时会执行 complete 回调。

04 播放语音接口：

```
wx.playVoice({
    localId: '' //需要播放的音频的本地 ID，由 stopRecord 接口获得
});
```

05 暂停播放接口：

```
wx.pauseVoice({
    localId: '' //需要暂停的音频的本地 ID，由 stopRecord 接口获得
});
```

06 停止播放接口:

```
wx.stopVoice({
  localId: '' //需要停止的音频的本地 ID, 由 stopRecord 接口获得
});
```

07 监听语音播放完毕接口:

```
wx.onVoicePlayEnd({
  success: function (res) {
    var localId = res.localId; //返回音频的本地 ID
  }
});
```

08 上传语音接口:

```
wx.uploadVoice({
  localId: '', //需要上传的音频的本地 ID, 由 stopRecord 接口获得
  isShowProgressTips: 1//默认为 1, 显示进度提示
  success: function (res) {
    var serverId = res.serverId; //返回音频的服务器端 ID
  }
});
```

09 下载语音接口:

```
wx.downloadVoice({
  serverId: '', //需要下载的音频的服务器端 ID, 由 uploadVoice 接口获得
  isShowProgressTips: 1//默认为 1, 显示进度提示
  success: function (res) {
    var localId = res.localId; //返回音频的本地 ID
  }
});
```

10 识别音频并返回识别结果接口:

```
wx.translateVoice({
  localId: '', //需要识别的音频的本地 ID, 由录音相关接口获得
  isShowProgressTips: 1, //默认为 1, 显示进度提示
  success: function (res) {
    alert(res.translateResult); //语音识别的结果
  }
});
```



备注: 语音智能接口可用于业务的快速输入。

5.6 ECharts 在微信中的应用

在微信 JSAPI 模式下,如果单纯地使用文字、数字、DIV、TABLE 等元素,那么界面风格将趋于复古化、固定化,在界面中适当的引入图表元素,可以大幅提高数据的可视化,增强用户体验。这里将展示 ECharts 在微信中的可视化应用,通过对本节的学习,使大家了解 ECharts,能够轻松将 ECharts 部署在微信中。

5.6.1 ECharts 简介

ECharts,缩写来自 Enterprise Charts,商业级数据图表,是由百度公司提供的-一个纯 JavaScript 的开源图表库,不仅可以流畅地运行在 PC 和移动设备上,而且兼容当前绝大部分浏览器 (IE6/7/8/9/10/11, Chrome, Firefox, Safari 等),底层依赖轻量级的 Canvas 类库 ZRender,提供直观、生动、可交互、可高度个性化定制的数据可视化图表。

支持折线图(区域图)、柱状图(条状图)、散点图(气泡图)、K 线图、饼图(环形图)、雷达图(填充雷达图)、和弦图、力导向布局图、地图、仪表盘、漏斗图、事件流程图 12 类图表,同时提供标题、详情气泡、图例、值域、数据区域、时间轴、工具箱 7 个可交互组件,支持多图表、组件的联动和混搭展现,同时创新地拖曳重计算、数据视图、值域漫游等特性大大增强了用户体验,赋予了用户对数据进行挖掘、整合的能力。

5.6.2 ECharts 快速接入

随着 ECharts 开源项目的不断升级,最新的 ECharts 3 不再强制使用 AMD 的方式按需引入,代码里也不再内置 AMD 加载器,所以正在使用 ECharts 3 之前版本的读者可以从烦琐的 AMD 引入中解脱出来,尝试使用 ECharts 3 版本,具体操作如下:

01 下载 ECharts JS 文件。

新版的 ECharts 3 文件体积更小,可以通过官方地址定制化下载,也可以通过 Github 下载,还可以通过 npm 获取 Echarts,详细地址如下:


- 官方网址: <http://echarts.baidu.com/download.html>
- Github 地址: <https://github.com/echarts>
- npm 获取: `npm install echarts --save`

02 引入 ECharts 文件。

ECharts 3 不再强制使用 AMD 的方式,所以文件的引入方式简单了许多,只需像普通的 JavaScript 库一样用 script 标签引入接口即可,完整示例代码如下:

```
<!DOCTYPE html>
<html>
<header>
  <meta charset="utf-8">
  <!-- 引入 ECharts 文件 -->
  <script src="echarts.min.js"></script>
```

```
</header>
</html>
```

 **备注：** <meta charset="utf-8">为页面编码格式，读者可以根据自己项目情况进行配置。

03 DEMO 示例创建。

为ECharts准备一个具备高宽的 dom 容器<div id="main" style="width: 600px;height:400px;"></div>，通过 echarts.init 方法初始化一个 echarts 实例，并通过 setOption 方法生成一个简单的柱状图，示例代码如下所示：

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>ECharts</title>
  <!-- 引入 echarts.js -->
  <script src="echarts.min.js"></script>
</head>
<body>
  <!-- 为ECharts准备一个具备大小（宽高）的 Dom -->
  <div id="main" style="width: 600px;height:400px;"></div>
  <script type="text/javascript">
    //基于准备好的 dom，初始化 echarts 实例
    var myChart = echarts.init(document.getElementById('main'));
    //指定图表的配置项和数据
    var option = {
      title: {
        text: 'ECharts 入门示例'
      },
      tooltip: {},
      legend: {
        data: ['销量']
      },
      xAxis: {
        data: ["一月", "二月", "三月", "四月", "五月", "六月"]
      },
      yAxis: {},
      series: [{
        name: '销量',
        type: 'bar',
        data: [5, 20, 36, 10, 10, 20]
      }]
    };
    //使用刚指定的配置项和数据显示图表
    myChart.setOption(option);
  </script>
```



```
</body>
</html>
```

示例结果如图 5.5 所示。

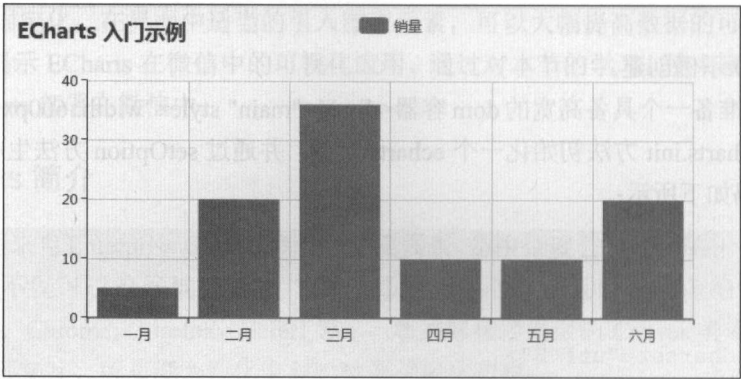


图 5.5 ECharts 快速接入 DEMO

5.6.3 ECharts 微信应用

ECharts 在微信中的应用可以让数据在移动端有更完美的展现，使数据能生动、可视化地展示，提高读者微信轻应用开发的视觉效果，如图 5.6 所示。

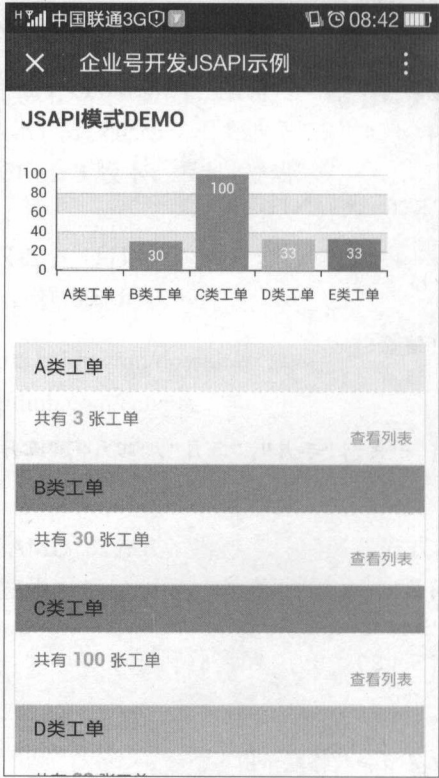


图 5.6 ECharts 微信应用

ECharts 微信应用完整示例代码如下:

```

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GBK">
<script type="text/javascript" src="jweixin-1.0.0.js"></script>
<script type="text/javascript" src="echarts.min.js"></script>
<meta name="viewport" content="width=device-width, initial-scale=1">
<script>
    wx.config({
        debug : true, //开启调试模式,调用的所有的 API 返回值会在客户端弹出消息提示,
//输出接口信息,若要查看传入的参数,可以在 PC 端打开,参数信息会通过 log 打印出来,仅在 PC
//端时才会打印
        appId : '${corpid}', //必填,企业号的唯一标识,此处填写企业号 CorpID
        timestamp : "${time}", //必填,生成签名的时间戳
        nonceStr : '${nonceStr}', //必填,生成签名的随机串
        signature : '${str1}', //必填,签名,见附录 A
        jsApiList : [ 'hideOptionsMenu' ]
//必填,需要使用的 JS 接口列表,所有 JS 接口列表见附录 B
    });
    wx.ready(function() {
        //config 信息验证后会执行 ready 方法,所有接口调用都必须在 config 接口获得结
//果之后,config 是一个客户端的异步操作,如果需要在页面加载时就调用相关接口,则必须把相关
//接口放在 ready 函数中调用来确保正确执行。对于用户触发时才调用的接口,则可以直接调用,无
//须放在 ready 函数中
        wx.hideOptionsMenu();
    });
    //列表展示
    function showBusiList(curtype){
        //跳转详细页面
    }
</script>
<title>企业号开发 JSAPI 示例</title>
<style type="text/css">
.titleTd {
    border-top: 1px solid #eeeeee;
    border-left: 1px solid #eeeeee;
    border-right: 1px solid #eeeeee;
    font-size: 16px;
    font-family: 宋体;
    padding-left: 15px;
}
.detailTd {
    border-top: 1px dashed #eeeeee;
    border-left: 1px solid #eeeeee;
    border-right: 1px solid #eeeeee;
    font-size: 14px;

```

```
font-family: 宋体;
padding-left: 15px;
}
</style>
</head>
<body style="background-repeat:no-repeat;background-position:center center">
  <table id="micromonthList" width="99%" style="table-layout: fixed;
border-collapse: collapse;" cellpadding="0" cellspacing="0">
    <tr>
      <td>
        <div id="main" style="width: 100%;height:200px;"></div>
<script type="text/javascript">
  //基于准备好的 dom, 初始化 Echarts 实例
  var myChart = echarts.init(document.getElementById('main'));
  var colorList = ['#f1f1f1','#5AB52C','#3AA6D9','#DDC768','#FF1111'];
  //指定图表的配置项和数据
  var option = {
    title: {
      text: 'JSAPI 模式 DEMO'
    },
    tooltip: {},
    xAxis: {
      axisLabel: {'interval':0},
      splitLine: { show: false },
      data: ["A 类工单","B 类工单","C 类工单","D 类工单","E 类工单"]
    },
    yAxis: {
      axisLine: { show: true },
      axisTick: { show: false },
      splitArea: {show:true}
    },
    series: [{
      name: '',
      type: 'bar',
      itemStyle:{
        normal: {
          label:{ show: true, position: 'insideTop' } ,
          color:function(params) {
            return colorList[params.dataIndex];
          }
        }
      },
      data: [3, 30, 100, 33, 33]
    }]
  };
  //使用刚指定的配置项和数据显示图表
```

```

        myChart.setOption(option);
    </script>
    </td>
</tr>
<tr height="40" bgcolor="# f1f1f1">
    <td class="titleTd"> A 类工单</td>
</tr>
<tr height="60">
    <td class="detailTd"><div style="color: #777777;">
        共有<span style="font-size: 16px; color: #FF9E2A; font-
weight: bolder;"> 3 </span>张工单
        </div>
        <div align="right" style="color: #1DB4F2; font-size: 13px;
padding-right: 10px" onclick="showBusiList('0')">查看列表</div></td>
</tr>
<tr height="40" bgcolor="#5AB52C">
    <td class="titleTd"> B 类工单</td>
</tr>
<tr height="60">
    <td class="detailTd"><div style="color: #777777;">
        共有<span style="font-size: 16px; color: #FF9E2A; font-
weight: bolder;"> 30 </span>张工单
        </div>
        <div align="right" style="color: #1DB4F2; font-size: 13px;
padding-right: 10px" onclick="showBusiList('1')">查看列表</div></td>
</tr>
<tr height="40" bgcolor="#4444BB">
    <td class="titleTd"> C 类工单</td>
</tr>
<tr height="60">
    <td class="detailTd"><div style="color: #777777;">
        共有<span style="font-size: 16px; color: #FF9E2A; font-
weight: bolder;"> 100 </span>张工单
        </div>
        <div align="right" style="color: #1DB4F2; font-size: 13px;
padding-right: 10px" onclick="showBusiList('2')">查看列表</div></td>
</tr>
<tr height="40" bgcolor="#DDC769">
    <td class="titleTd"> D 类工单</td>
</tr>
<tr height="60">
    <td class="detailTd"><div style="color: #777777;">
        共有<span style="font-size: 16px; color: #FF9E2A; font-
weight: bolder;"> 33 </span>张工单
        </div>
        <div align="right" style="color: #1DB4F2; font-size: 13px;

```



```
padding-right: 10px" onclick="showBusiList('3')">查看列表</div></td>
    </tr>
    <tr height="40" bgcolor="#FF1111">
        <td class="titleTd">E 类工单</td>
    </tr>
    <tr height="60">
        <td class="detailTd"><div style="color: #777777;">
            共有<span style="font-size: 16px; color: #FF9E2A; font-
weight: bolder;"> 33 </span>张工单
        </div>
        <div align="right" style="color: #1DB4F2; font-size: 13px;
padding-right: 10px" onclick="showBusiList('4')">查看列表</div></td>
    </tr>
    <tr>
        <td style="border-top: 1px solid #eeeeee;"></td>
    </tr>
</table>
</body>
</html>
```



使用技巧：<meta name="viewport" content="width=device-width, initial-scale=1">解决微信内置浏览器中的窗口大小问题。

5.7 微信中的地图语音导航

无论是服务号还是企业号，在微信开发过程中，地图都是最常见的客户需求之一，并且在各类地图的 API 中，都详细介绍了如何接入以及接口调用，本节将以腾讯地图、百度地图为例，讲解在微信中的语言导航功能。

5.7.1 微信内置地图导航

通过微信内置地图，实现微信查看以及导航，接口方法为 `wx.openLocation`，通过接口查看当前位置与目的地，接口说明如下：

```
wx.openLocation({
  latitude: 39.98871, //纬度，浮点数，范围为 90 ~ -90
  longitude: 116.43234, //经度，浮点数，范围为 180 ~ -180。
  name: '天安门', //位置名
  address: '中国北京东城区东长安街'//地址详情说明
  scale: 1, //地图缩放级别，整形值，范围从 1~28。默认为最大
  infoUrl: '' //在查看位置界面底部显示的超链接，可点击跳转
});
```

运行效果如图 5.7 所示。



图 5.7 腾讯地图语音导航

5.7.2 腾讯地图语音导航

微信属于腾讯旗下，所以在微信开发中，腾讯地图开发具有一定的优势。而建议在微信开发中使用的（其他地图也可以使用）腾讯地图的语音导航功能不仅支持在 App 中打开导航，而且还支持在没有地图 App 的情况下在微信中直接导航，如图 5.8 所示。



图 5.8 腾讯地图语音导航



备注：腾讯地图中的语音导航，不仅支持微信内直接导航，而且能打开腾讯地图 App 进行语音导航，除此之外，还支持打开其他地图 App 进行语音导航。

地图的导航功能在 Android、iOS 开发中提供的相应的接口，但是在手机 Web 开发（微信内置浏览器）中只能通过响应的 Web 服务实现。这里需要说明的是，如果使用腾讯地图提供 JavaScript API 是不存在导航的，只能通过 URI API 中获得，接口详细说明如下。

(1) 接口链接：

[http://apis.map.qq.com/uri/v1/routeplan?type=bus&from=家 &fromcoord=39.980,116.3&to=公司&tocoord=39.9,116.3&policy=1&referer=myapp](http://apis.map.qq.com/uri/v1/routeplan?type=bus&from=家&fromcoord=39.980,116.3&to=公司&tocoord=39.9,116.3&policy=1&referer=myapp)

(2) 参数说明如表 5.4 所示。

表 5.4 腾讯地图导航参数说明

参数	是否必需	说明	示例
type	是	路线规划方式参数： 公交：bus 驾车：drive 步行：walk（仅适用移动端）	type=bus type=drive type=walk
from	必填其一	起点名称	from=鼓楼
fromcoord		起点坐标 移动端如果起点名称和起点坐标均未传递，则使用当前定位位置作为起点	fromcoord=39.907380,116.388501
to	是	终点名称	to=奥林匹克森林公园
tocoord	否	终点坐标	tocoord=40.010024,116.392239
coord_type	否	坐标类型，取值如下： 1.GPS 2.腾讯坐标（默认） 如果用户指定该参数为非腾讯地图坐标系，则URI API自动进行坐标处理，以便准确对应到腾讯地图上	coord_type=1
policy	否	本参数取决于type参数的取值，公交： type=bus，policy有以下取值： 0：较快捷； 1：少换乘； 2：少步行； 3：不坐地铁。 驾车：type=drive，policy有以下取值： 0：较快捷； 1：无高速； 2：距离。 policy的取值默认为0	policy=1
referer	是	调用来源，一般为您的应用名称，为了保障对您的服务，请务必填写	referer=您的应用名称

备注：腾讯地图坐标转换可以查看 JavaScript API 中的 `convertor` 接口：

```
convertor.translate(points:LatLng | Point | Array.<LatLng> | Array.<Point>,type:Number, callback:Function)
```

将标准经纬度或其他地图经纬度转换为腾讯地图经纬度坐标，其中 `type` 的可选值为 1：GPS 经纬度；2：搜狗经纬度；3：百度经纬度；4：mapbar 经纬度；5：google 经纬度；6：搜狗墨卡托，代码如下：

```
qq.maps.convertor.translate(new qq.maps.LatLng(39.911082, 116.396135), 3,
function(res) {
    latlng = res[0];
    var marker = new qq.maps.Marker({
        map: map,
        position: latlng});
});
```

(3) 示例代码：

```
<a href="http://apis.map.qq.com/uri/v1/routeplan?type=drive&from= 我的家
&fromcoord=39.980683,116.302&to=中关村&tocoord=39.9836,116.3164&policy=1&referer=
myapp">腾讯地图 uri web 地图导航</a>
```

使用技巧：直接使用 `<a>` 标签调用即可，与 JavaScript API 不同，不需要在页面中引入 js 文件和开发者 key。

5.7.3 百度地图语音导航

百度地图的语音导航与腾讯地图不同，只能通过 App 进行导航，无法在微信中直接导航，如图 5.9 所示。



图 5.9 百度地图语音导航

与腾讯地图使用方式相同，百度地图的 JavaScript API 使用需要引入 JS 文件并获得 KEY 值，但微信中的语音导航使用的却是 URI API 接口，并不需要开发者 KEY，详细说明如下。

(1) 接口链接：

http://api.map.baidu.com/direction?origin=latlng:39.915,116.404|name:我家&destination=出发地的名字&mode=driving®ion=烟台市&output=html&src=appName

(2) 参数说明如表 5.5 所示。

表 5.5 百度地图导航参数说明

参数	是否必需	说明
origin	是	起点名称或经纬度，或者同时提供名称和经纬度，此时经纬度优先级高，将作为导航依据，名称只负责展示
destination	是	终点名称或经纬度，或者同时提供名称和经纬度，此时经纬度优先级高，将作为导航依据，名称只负责展示
mode	是	导航模式，固定为transit、driving、walking，分别表示公交、驾车和步行
region	当给定region时，认为起点和终点都在同一城市，除非单独给定起点或终点的城市	城市名或县名
origin_region		起点所在城市或县
destination_region		终点所在城市或县
output	是	表示输出类型，Web上必须指定为html才能展现地图产品结果
coord_type	否	坐标类型，可选参数
zoom	否	展现地图的级别，默认为视觉最优级别
src	是	appName




备注：origin 参数值的写法有三种：

- 1. 名称：天安门。
- 2. 经纬度：39.98871<纬度>,116.43234<经度>。
- 3. 名称和经纬度：name:天安门|latlng:39.98871,116.43234。

coord_type 的值默认为 bd09 经纬度坐标。允许的值为 bd09ll、bd09mc、gcj02 和 wgs84。bd09ll 表示百度经纬度坐标，bd09mc 表示百度墨卡托坐标，gcj02 表示经过国测局加密的坐标，wgs84 表示 GPS 获取的坐标。

(3) 示例代码：

```
<a href="http://api.map.baidu.com/direction?origin=latlng:39.915,116.404|name:我家&destination=东方&mode=driving&region=烟台市&output=html&src=appName">
  百度地图 uri web 地图导航
</a>
```


 **备注:** 直接调用 bdapp://map/direction?origin=latlng 的方式无法在微信内置浏览器中打开百度地图,但在手机浏览器中可以。

```
<a href="bdapp://map/direction?origin=latlng:34.264642646862,108.95108518068|name:我家
&amp;destination=目的地&amp;mode=driving&amp;region=烟台&amp;src=yourCompanyName|
yourAppName">bdapp 在微信内部浏览器不能导航</a>
```

5.8 微信 SPA 开发

这里的 SPA,可不是我们生活中的足疗、按摩,而是一种 Web 开发方式——Single Page Application (单页面应用),能够使用 AngularJS、Ember.js、onsenUI 等进行开发,不仅能够用来开发 Web-Android/iOS,而且能够开发微信。

单页面应用开发中,所有页面均可采用静态文件(.html),由主页面(index.html)进入,通过 push、replace 以及 pop 的形式加载页面,实现页面跳转。在页面数据加载上,采用 Ajax 等形式加载数据。本节将通过 onsenUI、AngularJS 带领读者学习 SPA 下如何进行微信开发。

 **备注:** onsenUI 是 AngularJS 基础上封装的单页面应用框架,读者可以通过 <https://onsen.io/v1/reference/javascript.html> 学习。对 SPA 感兴趣的读者可以查看 onsenUI 和 AngularJS,本节主要讲解 onsenUI 和 AngularJS 如何实现微信开发。

5.8.1 基于 AngularJS 的 onsenUI

onsenUI 中的基础语法为 AngularJS,标签格式为<ons-***>,与普通的 struts 开发中的 struts 标签用法较为相似,AngularJS 表达式则与 EL 表达式相似。EL 表达式为\${user.name},而 AngularJS 表达式为{{user.name}},通过 html 中的{{ user.name }}能够实现主页 index.html,示例代码如下:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="lib/onsen/css/onsenui.css"/>
    <link rel="stylesheet" href="lib/onsen/css/onsen-css-components.css"/>
    <script src="lib/onsen/js/angular/angular.js"></script>
    <script src="lib/onsen/js/onsenui.js"></script>
    <script>
      var module = ons.bootstrap('my-app', ['onsen']);
      module.controller('AppController', function($scope) { });
      module.controller('PageController', function($scope) {
        ons.ready(function() {
          // Init code here
        });
      });
    </script>
```

```

</script>
</head>
<body ng-controller="AppController">
  <ons-navigator var="navigator">
    <ons-page ng-controller="PageController">
      <!-- Page content -->
    </ons-page>
  </ons-navigator>
</body>
</html>

```

onsenUI 除了<ons-tabbar>实现页面跳转之外（如图 5.10 所示），还提供了<ons-navigator>标签，为 onsenUI 中的导航控制标签，可以通过以下方式加载页面：

```

navigator.pushPage('page2.html');//加载新页面，原页面放入堆栈
navigator.pushPage("page2.html", options);
navigator.popPage();//返回堆栈中上一页面，pop 当前页面
navigator.replacePage('page2.html');//加载新页面，原页面不放入堆栈
navigator.replacePage ("page2.html", options);

```



备注：options 为 JSON 对象，如 { param1: "value1", param2: "value2", animation: 'slide' }, 其中 animation 为页面加载时的动画效果，param1、param2 为传递到下一页面的参数。

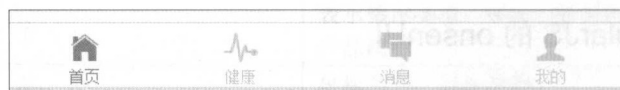


图 5.10 onsenUI 中的<ons-tabbar>标签

5.8.2 创建 AngularJS 微信服务

单页面应用通过主页一次性加载所有 JS 文件，所有对于公用的微信服务文件可以使用 AngularJS 中的 service 服务创建新的微信服务，示例代码如下，创建公用方法“判断是否是微信浏览器”和“获取 URL 中的参数”：

```

module.factory('$wxUtil',[ '$http',function($http) {
  var $s = {};
  //判断是不是微信浏览器
  $s.isWeixin = isWeixin;
  function isWeixin(){
    var ua = window.navigator.userAgent.toLowerCase();
    if(ua.match(/MicroMessenger/i)!="micromessenger") {
      window.location.href="safepage.html";
    }
  }
  //argName 表示要获取哪个参数的值
  $s.getArgsFromHref = getArgsFromHref;

```

```

function getArgsFromHref(argName){
    //测试"Untitled-2.html?id=2"
    var sHref = window.location.href.split("#")[0];
    var args = sHref.split("?");
    var retval = "";
    //参数为空
    if(args[0] == sHref) {
        return retval; /*无须做任何处理*/
    }
    var str = args[1];
    args = str.split("&");
    for(var i = 0; i < args.length; i++ ) {
        str = args[i];
        var arg = str.split("=");
        if(arg.length <= 1) continue;
        if(arg[0] == sArgName) retval = arg[1];
    }
    return retval;
}
return $s;
});

```

5.8.3 SPA 下 JSAPI 模式权限初始化

单页面应用中的数据需要通过异步推送访问数据，并且 URL 链接不变（单页面应用只变#后面部分），所以微信 JS-SDK 的授权页面授权 SPA 的主页面即可，通过 window.location 获得主页面地址，示例代码如下：

```
var $$$url=window.location.href.split("#")[0];
```

将主页地址传入后台，由后台利用有效的 jsapi_ticket、noncestr（随机字符串）、timestamp（时间戳）和 URL（单页面应用主页地址，不包含#及其后面部分）四个变量完成权限验证签名，获得签名 signature，将 signature、corpId、timestamp 和 noncestr 组成 JSON 数据传递至前台，由前台进行解析、初始化微信 wx.config() 方法，示例代码如下：

```

//获得主页链接，不包括#及其后面部分
var $$$url=window.location.href.split("#")[0];
//将主页 URL 传至后台，获取权限验证签名
$.http({
    url: '/wx/getWxPageConfig.do',
    method: 'POST',
    data: {curUrl: $$$url }
}).success(function(){
    //链接成功获得数据，初始化微信 JS-SDK 授权签名
    var errcode=data.errcode;
    var errmsg=data.errmsg;


```



```


wx.config({
  debug: false, //开启调试模式,调用的所有API的返回值会在客户端弹出消息提
//示,输出接口信息,若要查看传入的参数,可以在PC端打开,参数信息会通过log打印出来,仅在
//PC端时才能打印
  appId: data.appid, //必填,企业号的唯一标识,此处填写企业号 CorpID
  timestamp: data.time, //必填,生成签名的时间戳
  nonceStr: data.nonceStr, //必填,生成签名的随机串
  signature: data.str1, //必填,签名,见附录A
  jsApiList: ['hideOptionsMenu', 'getLocation', 'chooseImage',
'hideAllNonBaseMenuItem'] //必填,需要使用的JS接口列表,所有JS接口列表见附录B
});
wx.ready(function(){
  //微信回调方法
  //获得坐标
  wx.getLocation({
    success: function (res) {
      var latitude = res.latitude; //纬度,浮点数,范围为 90 ~ -90
      var longitude = res.longitude; //经度,浮点数,范围为 180 ~ -180。
      var speed = res.speed; //速度,以米/每秒计
      var accuracy = res.accuracy; //位置精度
      //console.log("-----地理位置----: ", res);
      $scope.posi=res;
    }
  });
});
}).error(function(){
});

```

 **备注:** 后台获取签名的方法请参照“5.2.3 权限验证”，获取的账户信息可以存放至 LocalStorage 中。

5.8.4 SPA 下获取 OAuth 2.0 成员身份信息

SPA 中所有页面均通过入口 index 页面进入,因此 OAuth 2.0 链接(将在 8.1 节介绍)的处理需要对主页面进行处理,通过主页获取微信 code,将 code 传递至后台获取成员身份信息,并保存至 session 和前台页面中,处理方式如下。

 **备注:** 由于 OAuth 2.0 不支持 push 特性,也不支持 Ajax,所以只能通过主页获取 code,进而得到成员 uesId。缺陷是,将 userId 存至前台,利用前台 userId 操作时具有一定的安全隐患。

01 对微信菜单链接进行处理,并指向单页面主页 wxAccountIndex.html,示例代码如下:

```
https://open.weixin.qq.com/connect/oauth2/authorize?appid=wx12345789ffff
```

```
f3c59&redirect_uri=http%3a%2f%2fmyfmyfmyfmyf.vicp.cc%2fWXDEMO%2fwx%2fwxAccountIndex.html&response_type=code&scope=snsapi_userinfo&state=123#wechat_redirect
```


02 通过 OAuth 2.0 链接获得身份链接，链接如下：

```
http://myfmyfmyfmyf.vicp.cc/WXDEMO/wx/wxAccountIndex.html?code=041Mkpus0Zdsrr1Uoiss0d6uus0MkpuY&state=123#eyJpbmRleCI6MiwicGFnZSI6Ii4uL3d3dy9wZXJzb25hbC9wZXJzb25hbC5odGlsIn0=
```

 备注：#号部分为 SPA 页面的 HASH 信息。

03 从主页控制器 controller 中获取 code 信息，将 code 信息通过 AngularJS 中的 \$http 服务传递至后台，由后台通过 code 向企业号换取成员信息，示例代码如下：

```
//获取 URL 地址中 code
var $$$code=$wxUtil.getArgsFromHref("code");
//获得 URL 地址
var $$$url=window.location.href.split("#")[0];
```

 备注：\$wxUtil 为自定义服务，在 5.8.2 节中介绍。code 获取成员信息将在 8.1.2 节中介绍。URL 链接中#及#后面部分不参与权限授权签名。

5.8.5 解决微信物理回退

物理回退，指通过手机按键中的“回退”键实现的页面后退功能。在 SPA 开发中，由于 URL 地址无变化，导致微信中点击“回退”键直接退出微信内置浏览器，那么如何解决 SPA 中的物理回退问题呢？window.location.hash 通过创建监听事件 postpush（页面加载之后），修改 URL 链接中的 hash 值，使 URL 地址发生变化（由于 hash 值为#值，所以不影响微信 JS-SDK 授权），从而解决回退问题。以 onsenUI 为例，示例代码如下：

```
//调用回退
setImmediate(function () {parseHashChange(mainNavi)});
/**
 * 物理回退
 * @param mainNavi
 */
function parseHashChange(mainNavi){
  if(!mainNavi) return;
  mainNavi.on('postpush', function(event) {
    //监听 push 后事件
    //{index: 3, page: "../www/login/login.html"}
    var pages = event.navigator.pages;
    var pageName = event.enterPage.page;
    console.log("pages:", pages);
    console.log("pageName:", pageName);
```

```

        var hash = {index:pages?pages.length:0,page:pageName};
        console.log("hash:",hash);
        window.location.hash = window.btoa(angular.toJson(hash));
    });
    mainNavi.on('prepush', function(event) {
        //监听 push 之前的事件
    });
    mainNavi.on('prepop', function(event) {
        //监听 pop 之前的事件
    });
    mainNavi.on('postpop', function(event) {
        //监听 pop 之后的事件
    });
    angular.element(window).on('hashchange', function(event){
        //监听 hashchange 事件
    });
}

```



备注：hash 属性是一个可读可写的字符串，是 URL 的锚部分（从#号开始的部分），可以通过 window.location.hash 修改 hash 值；window.btoa 和 window.atob 是 JS 中的原生方法，用于进行 Base64 转码和解码。

5.9 微信 WebSocket 开发

WebSocket 是 H5 中的新特性，企业号中也支持 WebSocket 的开发。与普通 Socket 开发相同，WebSocket 也需要客户端和服务端的支持，客户端为浏览器页面，服务端为后台服务支撑。就像水管与水一样，“水管”是 WebSocket 通道，一头连接在用户水龙头，一头连接在蓄水池，“用户水龙头”犹如 WebSocket 客户端，“蓄水池”则是 WebSocket 服务端，水则是 WebSocket 中的数据。

5.9.1 WebSocket 客户端

WebSocket 客户端为浏览器页面，只需在 JS 中使用 new WebSocket 便可以在客户端开通，示例代码如下：

```

<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%
    String path = request.getContextPath();
    String basePath = request.getScheme()+ "://" + request.getServerName() + ":" +
request.getServerPort() + path + "/";
%>
<!DOCTYPE HTML>
<html>
<head>

```



```
//关闭连接
function closeWebSocket(){
    websocket.close();
}
//发送消息
function send(){
    var message = document.getElementById('text').value;
    websocket.send(message);
}
</script>
</body>
</html>
```



注意：如果使用 WebSocket，读者在开通 http 服务的同时，需要开通 WS（WebSocket）协议的服务。WS 协议类似 TCP 协议、HTTP 协议等，写法：`ws://myfmyfmyfmyf.vicp.cc/SELearning/ws/chat/33`。

5.9.2 WebSocket 服务端

后台服务需要导入工程中需要导入 `websocket-api.jar` 的包，该 jar 包可以在 Tomcat 的 lib 库中获得。导入成功后，便可以使用 `ServerEndpoint` 注解指定 URI，在无须配置 `web.xml` 的情况下，客户端便可以通过注解中的 URI 连接到 WebSocket，示例代码如下：

```
package com.service;
import java.util.HashMap;
import java.util.Map;
import javax.websocket.OnClose;
import javax.websocket.OnError;
import javax.websocket.OnMessage;
import javax.websocket.OnOpen;
import javax.websocket.Session;
import javax.websocket.server.PathParam;
import javax.websocket.server.ServerEndpoint;
@ServerEndpoint(value = "/ws/chat/{client-id}")
public class WebSocketService {

    private static Map<String, Session> clients = new HashMap<String, Session>();


    @OnOpen
    public void onOpen(Session client, @PathParam("client-id") String clientId) {
        System.out.println("用户" + clientId + "接入");
        clients.put(clientId, client);
    }
}
```

```

    @OnMessage
    public void onMessage(String message, @PathParam("client-id") String
clientId) {

    }
    @OnClose
    public void onClose(Session session, @PathParam("client-id") String
clientId) {
        System.out.println("向用户" + clientId + "关闭连接");
        clients.remove(clientId);
    }
    @OnError
    public void onError(Session session, Throwable error, @PathParam
("client-id") String clientId){
        clients.remove(clientId);
        error.printStackTrace();
    }
    public static void sendMessage(String message, String clientId) throws
Exception{
        if(clients.containsKey(clientId)){
            System.out.println("向用户" + clientId + "发送消息: " + message);
            clients.get(clientId).getBasicRemote().sendText(message);
        }
    }
}

```

 **备注：**使用 WebSocket 时需要注意连接数量以及稳定性（心跳）的问题。例如，I5，8GB 内存 64 位的普通台式机在默认配置的 Tomcat 7.0 中能够成功连接的 WebSocket 数量为 256 个。读者可以自行编写检测（测试时可能会造成机器无法运行），示例代码如下：

```

<input type="button" onclick="wbNumClick()" value="测试连接数:" /><span
id="wbNum">1</span>
<script type="text/javascript">
function wbNumClick(){
    var wbNum = document.getElementById('wbNum').innerHTML;
    var i=0;
    while(1==1){
        i=parseInt(i)+1;
        var websocket2 = new WebSocket("ws://localhost:8089/WSCON/ws/chat/"+i);
        websocket2.onopen = function(event){
            console.log(i+"接入");
        };
        websocket2.onclose = function(){
            console.log(i+"关闭");
        };
        document.getElementById('wbNum').innerHTML=i;
    }
}

```

```
}  
}  
</script>
```

5.10 微信中的支付宝

众所周知，微信支付与支付宝支付是对立的平台，如果客户需要在微信中支持支付宝，那麽效果又是如何呢，本节将演示如何在微信中使用支付宝。

首先，通过支付宝手机网站支付，完成签名，发起支付之后，页面将提示“如需浏览，请长按网址复制后使用浏览器访问”，如图 5.11 所示，读者可以按照提示进行操作，也可以点击右上角按钮“在浏览器中打开”。

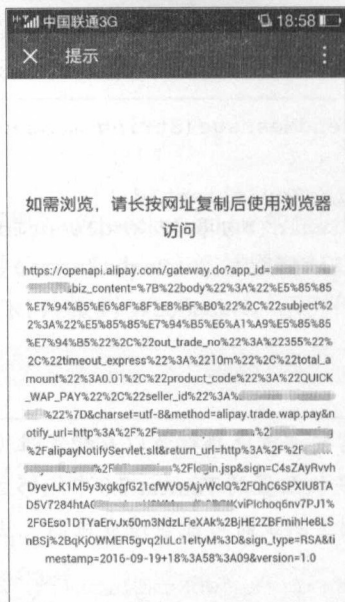


图 5.11 在微信中打开支付宝支付

支付宝手机网站支付发起页面示例代码如下：

```
// 支付页面  
// H5 支付页面——测试连接  
mainNavi.pushPage("https://openapi.alipaydev.com/gateway.do?" + data.signStr);  
// H5 支付页面——正式连接  
mainNavi.pushPage("https://openapi.alipay.com/gateway.do?" + data.signStr);
```



备注：data.signStr 为支付宝支付签名，经过 urlencode 处理。

通过浏览器打开支付便可以正常完成支付宝支付，如图 5.12 所示。

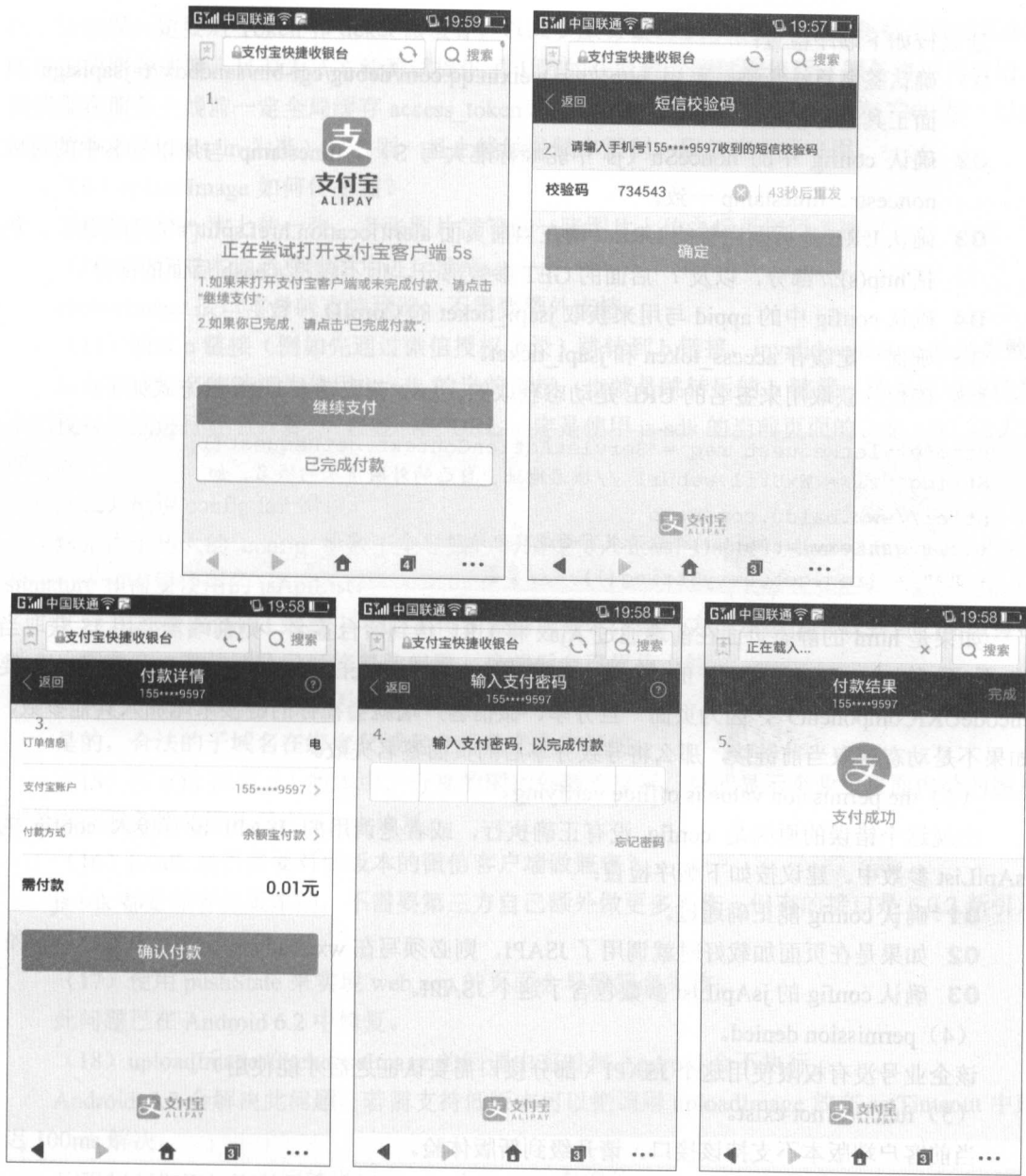


图 5.12 手机浏览器完成支付宝支付

5.11 常见问题

调用 config 连接的时候传入参数 debug: true, 可以开启 debug 模式, 页面会弹出消息提示, 输出错误信息。下面为常见错误及解决方法。

- (1) invalid url domain.
当前页面所在域名与使用的 CorpID 没有绑定(可在该企业号的应用可信域名中配置域名)。
- (2) invalid signature 签名错误。

建议按如下顺序检查：

- 01** 确认签名算法正确，可用 `http://mp.weixin.qq.com/debug/cgi-bin/sandbox?t=jsapisign` 页面工具进行校验。
- 02** 确认 `config` 中的 `nonceStr`（js 中驼峰标准大写 S）、`timestamp` 与用以签名中的对应 `noncestr`、`timestamp` 一致。
- 03** 确认 URL 是页面完整的 URL（请在当前页面 `alert(location.href.split('#')[0])` 确认），包括 `'http(s)://'` 部分，以及 `'?'` 后面的 GET 参数部分，但不包括 `#hash` 后面的部分。
- 04** 确认 `config` 中的 `appid` 与用来获取 `jsapi_ticket` 的 `CorpID` 一致。
- 05** 确保一定缓存 `access_token` 和 `jsapi_ticket`。
- 06** 确保你获取用来签名的 URL 是动态获取的，Java 获取动态 URL 的方式如下：

```
HttpServletRequest req = ServletActionContext.getRequest();
String url = WxUtil.webUrl //域名地址，自己的外网请求的域名，如
http://www.baidu.com/Demo
+ req.getServletPath() //请求页面或其他地址
+ "?" + (req.getQueryString()); //参数
```

如果是 html 的静态页面在前端通过 Ajax 将 URL 传到后台签名，则前端需要用 JS 获取当前页面除去 `#hash` 部分的链接（可用 `location.href.split('#')[0]` 获取，而且需要 `encodeURIComponent`），因为页面一旦分享，微信客户端就会在你的链接末尾加入其他参数，如果不是动态获取当前链接，那么将导致分享后的页面签名失败。

(3) the permission value is offline verifying.

出现这个错误的原因是 `config` 没有正确执行，或者是调用的 JSAPI 没有传入 `config` 的 `jsApiList` 参数中。建议按如下顺序检查：

- 01** 确认 `config` 能正确通过。
- 02** 如果是在页面加载好时就调用了 JSAPI，则必须写在 `wx.ready` 的回调中。
- 03** 确认 `config` 的 `jsApiList` 参数包含了这个 JSAPI。

(4) permission denied.

该企业号没有权限使用这个 JSAPI（部分接口需要认证之后才能使用）。

(5) function not exist.

当前客户端版本不支持该接口，请升级到新版体验。

(6) 为什么 6.0.1 版本 `config` 能正确通过，但是 6.0.2 版本之后就不能正确通过了呢。

因为 6.0.2 版本之前没有做权限验证，所以 `config` 都是能正确通过的，但这并不意味着你的 `config` 中的签名是对的，请在 6.0.2 版本中检验是否生成正确的签名，以保证 `config` 在高版本中也能正确通过。

(7) 在 iOS 和 Android 都无法分享。

请确认企业号已经认证，只有认证的企业号才具有分享相关接口的权限，如果确实已经认证过，则要检查监听接口是否在 `wx.ready` 回调函数中触发。

(8) 服务上线之后无法获取 `jsapi_ticket`，但自己测试时没问题。

因为 `access_token` 和 `jsapi_ticket` 必须要在自己的服务器中缓存，否则上线后会触发频率限

制。请确保一定要对 Token 和 ticket 做缓存，以减少服务器请求，不仅可以避免触发频率限制，还能加快服务速度。目前为了方便测试提供了 1 万的获取量，超过阈值后，服务将不再可用。请确保在服务上线前一定全局缓存 access_token 和 jsapi_ticket，两者有效期均为 7200 秒（以返回结果中的 expires_in 为准），否则一旦上线触发频率限制，服务将不再可用。

(9) uploadImage 如何传多图？

目前只支持一次上传一张，多张图片需等前一张图片上传之后再调用该接口。

(10) 没法对本地选择的图片进行预览。

chooseImage 接口本身就支持预览，不需要额外支持。

(11) 通过 a 链接（例如先通过微信授权登录）跳转到 b 链接，invalid signature 签名失败。

后台生成签名的链接为使用 js-sdk 的当前链接，也就是跳转后的 b 链接，请不要用微信登录的授权链接进行签名计算，后台签名的 URL 一定是使用 js-sdk 的当前页面的完整 URL 除去 # 部分。

(12) 出现 config:fail 错误。

这是由于传入的 config 参数不全导致，请确保传入正确的 appId、timestamp、nonceStr、signature 和需要使用的 jsApiList。

(13) 之前 Android 通过 js-sdk 上传到微信服务器，第三方再从微信下载到自己的服务器时会出现杂音，微信团队已经修复此问题，目前后台已优化上线。

(14) 绑定父级域名，是否其子域名也是可用的？

是的，合法的子域名在绑定父域名之后是完全支持的。

(15) 在 iOS 微信 6.1 版本中，分享的图片外链不显示，只能显示企业号页面内链的图片或者微信服务器的图片已在 6.2 中修复。

(16) js-sdk 是否需要低版本的微信客户端做兼容？

js-sdk 都是兼容低版本的，不需要第三方自己额外做更多工作，但有的接口是 6.0.2 新引入的，只有新版才可调用。

(17) 使用 pushState 来实现 web app 的页面会导致签名失败。

此问题已在 Android 6.2 中修复。

(18) uploadImage 在 chooseImage 的回调中有时 Android 会不执行。

Android 6.2 会解决此问题，若需支持低版本可以把调用 uploadImage 放在 setTimeout 中延迟 100ms 解决。

(19) getLocation 返回的坐标在 openLocation 有偏差。

因为 getLocation 返回的是 GPS 坐标，openLocation 打开的腾讯地图为火星坐标，需要第三方自己做转换，6.2 版本已开始支持直接获取火星坐标。

(20) 点击数字拨打电话：

```
<a href="tel:${book.mobile}">
```

(21) 点击数字发送短信：

```
<a href="sms:${book.mobile}">
```

(22) 出现 redirect_uri 错误时，需

01 检查连接中的 redirect_uri 参数值是否进过 urlencode 编码？

02 是否添加可信域名？

03 非 80 端口的可信域名是否添加端口号？

5.12 案例：现场业务上报

通过对 JSAPI 模式的学习相信大家已经了解微信 JS-SDK 接口的注入和使用了，接下来我们就来实际练习一下微信接口的应用——现场业务上报。通过本节案例的展示，大家可以更好地学习权限接口的签名认证，以及如何使用微信 JS-SDK 接口。

5.12.1 场景回顾

对于服务类企业公司，客户需要经常上报抢修、新装、维护等业务，如客户预约燃气公司开通燃气、客户预约电力企业维护电表等，需要通过微信将工单直接派发给现场业务人员，并需要业务人员及时上报业务处理情况，对于需要上传现场实施图片的工单，也可以直接上报现场处理情况，与 PC 端（电脑登录端）录入相比，处理业务更及时，业务人员也更方便。

5.12.2 示例代码展示

界面功能展示如图 5.13 所示，这里布局及特效效果可以根据自身情况进行完善，可以增加页面背景图片或使用 HTML 5 新特性进行改善。

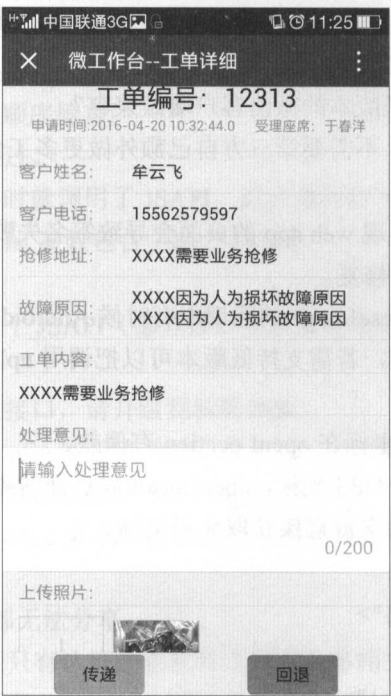


图 5.13 案例现场业务上报

这里我们新建一个 JSP 页面，命名为 Demo.jsp，在 Demo.jsp 中使用微信 JS-SDK 接口，示

例代码如下所示:

```
<%@ page language="java" contentType="text/html; charset=GBK" pageEncoding=
"GBK"%>
<%@ include file="../../../framework/include/pageset.jspa"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GBK">
<link rel="stylesheet" href="<%=request.getContextPath()%>/wx/css/weui.css"/>
<link rel="stylesheet" href="<%=request.getContextPath()%>/wx/css/
example.css"/>
<script type="text/javascript" src="<%=request.getContextPath()%>/wx/js/
jweixin-1.0.0.js"></script>
<%
//识别微信浏览器
String userAgent=request.getHeader("User-Agent");//里面包含了设备类型
if(-1==userAgent.indexOf("MicroMessenger")){
    //如果不是微信浏览器,则跳转到安全页
    request.getRequestDispatcher("noRightPage.jsp").forward
(request, response);
}
%>
<meta name="viewport" content="width=device-width, initial-scale=1">
<script>
    wx.config({
        debug: false, // 开启调试模式,调用的所有的 API 返回值会在客户端弹出消息提示,
//输出接口信息,若要查看传入的参数,可以在 PC 端打开,参数信息会通过 log 打印出来,仅在 PC
//端时才能打印
        appId: '${corpid}', //必填,企业号的唯一标识,此处填写企业号 CorpID
        timestamp: '${time}', //必填,生成签名的时间戳
        nonceStr: '${nonceStr}', //必填,生成签名的随机串
        signature: '${str1}', //必填,签名,见附录 A
        jsApiList:
['hideOptionsMenu','chooseImage','uploadImage','checkJsApi'] //必填,需要使用的
//JS 接口列表,所有 JS 接口列表见附录 B
    });
    wx.ready(function(){
        //config 信息验证后会执行 ready 方法,所有接口调用都必须在 config 接口获得结
//果之后,config 是一个客户端的异步操作,如果需要在页面加载时就调用相关接口,则必须把相关
//接口放在 ready 函数中调用出来以确保正确执行。对于用户触发时才调用的接口,则可以直接调用,
//不需要放在 ready 函数中。
        wx.hideOptionsMenu();
    });
    //选择图片
    var imgId=0;
    function loadImage(){
        wx.chooseImage({
```



```

        count: 9, //默认 9
        sizeType: ['original', 'compressed'], //可以指定是原图还是压缩图,
//默认二者都有
        sourceType: ['album', 'camera'], //可以指定来源是相册还是相机, 默认
//二者都有
        success: function (res) {
            var localIds = res.localIds; //返回选定照片的本地 ID 列表, localId
//可以作为 img 标签的 src 属性显示图片
            var tempHtml="";
            for (var i=0; i < localIds.length; i++) {
                tempHtml+="

```

```

        //跳转到传递 action 或者 servlet 进行处理
        $("#action").val("busiTrans");
        $("form")[0].submit();
    }
</script>
<title>微工作台--工单详细</title>
<style type="text/css">
p{
    font-family:宋体;
    font-size:12px;
    font-weight:normal;
}
</style>
</head>
<body
style="overflow:hidden;padding:0px;margin:0px;background-color:#eeeeff">
    <s:form action="wxBusinessAction" namespace="/wx" id="myform">
        <table width="100%" height="100%" style="table-layout:fixed"
cellpadding="0" cellspacing="0">
            <tr>
                <td colspan="2">
                    <div style="height:100%;width:100%;overflow:auto;">
                        <p >
                            工单类型: ${busi.busiType} <div style="width:100%;font-size:
24px;color:#000000;" align="center"> 工单编号: ${busi.busiId}</div>
                        </p>
                        <p>
                            <div
style="width:100%;height:30;font-size:13px;color:#999999;margin-top:5px"
align="center">
                                <span style="padding-left:5px;">申请时间: ${busi.appTime}</span>
                                <span style="padding-left:15px;">受理座席: ${busi.servAgent}
</span>
                            </div>
                        </p>
                        <div class="weui_cell">
                            <div class="weui_cell_hd"><label class="weui_label" style=
"color:#999999;"> 客户姓名: </label></div>
                            <div class="weui_cell_bd weui_cell_primary"> ${busi.customerName}
</div>
                        </div>
                        <div class="weui_cell">
                            <div class="weui_cell_hd"><label class="weui_label" style=
"color:#999999;"> 客户电话: </label></div>
                            <div class="weui_cell_bd weui_cell_primary"> ${busi.tel}</div>
                        </div>
                    </div>
                </td>
            </tr>
        </table>
    </s:form>

```

```

        <s:if test="#request.busi.repairAddress!=null">
            <div class="weui_cell">
                <div class="weui_cell_hd"><label class="weui_label"
style="color:#999999;"> 抢修地址: </label></div>
                <div class="weui_cell_bd weui_cell_primary">${busi.servContext}
</div>
            </div>
        </s:if>
        <s:if test="#request.busi.repairAddress!=null">
            <div class="weui_cell">
                <div class="weui_cell_hd"><label class="weui_label"
style="color:#999999;"> 故障原因: </label></div>
                <div class="weui_cell_bd weui_cell_primary">${busi.repairResion}
</div>
            </div>
        </s:if>
        <div class="weui_cell">
            <label class="weui_label" style="color:#999999;"> 工单内
容: </label>
        </div>
        <div class="weui_cell" style="margin-top:-8px">
            <div class="weui_cell_bd weui_cell_primary">${busi.
servContext}</div>
        </div>
        <s:if test="( #request.busi.accessFile!=null) && ( #request.busi.
accessFile.size() !=0)">
            <p
style="font-size:18px;color:#999999;line-height:150%;">
                附件照片: <s:iterator value="#request.busi.accessFile" var=
"item" status="st">
                    <img src='wxBusinessAction.do?action=
getImageFile&accessId=<s:property value="%{#item}" />' id='"+imgId+"
height='80px' width='80px' />
                </s:iterator>
            </p>
        </s:if>
        <s:if test="( #request.busi.dealStatus=='0'.toString()) ||
( #request.busi.dealStatus=='1'.toString())">
            <div class="weui_cell">
                <label class="weui_label" style="color:#999999;"> 处理意
见: </label>
            </div>
            <div class="weui_cells weui_cells_form" style="margin-top:
-8px">
                <div class="weui_cell">
                    <div class="weui_cell_bd weui_cell_primary">

```



```

        <textarea class="weui_textarea" name="dealOption"
id="dealOption" placeholder="请输入处理意见" rows="3"></textarea>
        <div
class="weui_textarea_counter"><span>0</span>/200</div>
        </div>
    </div>
    <div class="weui_cell">
        <label class="weui_label" style="color:#999999;"> 上传照片: </label>
    </div>
    <div class="weui_cell" style="margin-top:-8px">
        <a href="javascript:loadImage()" class="weui_
uploader_input_wrp" ></a><br/>
        <div id="imageList">
        </div>
    </div>
</s:if>
</div>
</td>
</tr>
<tr height="40">
    <td valign="middle" align="center" >
        <a id="transButton" href="javascript:void();" onclick=
"javascript:busiTrans();" class="weui_btn weui_btn_primary" style="width:
48%;border-top:1px solid #eeeeee;border-left:1px solid #eeeeee;border-right:
1px solid #eeeeee;color:#000000;font-size:16px;visibility: hidden" valign=
"middle" align="center">&nbsp;传递&nbsp;</a>
    </td>
    <td valign="middle" align="center" >
        <a id="backButton" href="javascript:void();" onclick=
"javascript:busiBack();" class="weui_btn weui_btn_warn" style="width:48%;
border-top:1px solid #eeeeee;border-left:1px solid #eeeeee;border-right:1px
solid #eeeeee;color:#000000;font-size:16px;visibility: hidden" valign="middle"
align="center">&nbsp;回退&nbsp;</a>
    </td>
    <input type="hidden" id="curBusiId" name="curBusiId" value=
"${busi.busiId}" />
    <input type="hidden" id="curFlowId" name="curFlowId" value=
"${busi.busiFlowId}" />
    <input type="hidden" id="addressUrl" name="addressUrl" value=
"${addressUrl}" />
    <input type="hidden" id="curUserId" name="curUserId" value=
"${curUserId}" />
    <input type="hidden" id="curState" name="curState" value=
"${busi.dealStatus}" />

```



```

        <input type="hidden" id="wxImageMediaId" name="wxImageMediaId"
value="" />
        <input type="hidden" id="action" name="action" value="" />
    </tr>
</table>
</s:form>
</body>
</html>

```

通过本章的学习我们知道，JS-SDK 平台的使用需要进行权限签名，这里需要在 action 或者 servlet 中进行权限签名，代码如下：

```


/**
 * 查看工单详细
 * @return
 */
public String showDetail(){
    HttpServletRequest req = ServletActionContext.getRequest();
    String url = WxUtil.webUrl          //域名地址，外网请求的域名
        + req.getServletPath()          //请求页面或其他地址
        + "?" + (req.getQueryString()); //参数
    System.out.println(url);
    //识别微信浏览器
    String userAgent=req.getHeader("User-Agent");//里面包含了设备类型
    if(-1==userAgent.indexOf("MicroMessenger")){
        //如果不是微信浏览器，则跳转到安全页
        return "safePage";
    }
    //获得 OAuth 2.0 验证
    String code=req.getParameter("code");
    String state=req.getParameter("state");
    //根据 code 获得人员
    WxUtil msgUtil=new WxUtil();
    String userId=msgUtil.getUserIdByCode(code);
    if(null==userId||"".equals(userId)||"null".equals(userId)){
        return "noRightPage";
    }
    System.out.println(userId+" 查看了工单 "+busiId);
    //-----
    //这里读者根据自己的业务接口查询出工单详细
    //然后通过 req.setAttribute("busi", busi);将实体类存入 request 中
    //-----
    //生成微信 JS 授权
    String jsapi_ticket=msgUtil.getJsapiTicketFromWx();//签名
    //通过动态 URL 生成签名
    Map<String, String> ret = WxUtil.sign(jsapi_ticket, url);
    req.setAttribute("str1", ret.get("signature"));//权限配置—签名

```

```

req.setAttribute("time", ret.get("timestamp")); // 权限配置——时间戳
req.setAttribute("nonceStr", ret.get("nonceStr")); // 权限配置——随机字符串
req.setAttribute("corpid", WxUtil.RESP_MESSAGE_CORPID); // 权限配置——CprID
req.setAttribute("addressUrl", WxUtil.webUrl); // 外网域名地址
// 获取展示的信息
String busiType = busi.getBusiType(); // 工单类型, 根据类型不同可以跳转到不同
// 页面, 显示不同的界面元素
req.setAttribute("busi", busi); // 存入工单数据
// struts 跳转页面
return "consoleDetail";
}

```

 **备注:** request.getHeader("User-Agent")里包含了设备类型, 可用于数据安全展示, 详细介绍将在第8章中讲解。

第 6 章

企业会话模式

企业会话模式，是与企业内部 IM 软件相结合的软件，实现企业内部 IM 软件与微信进行信息传递的方式。通过本章的学习将使读者掌握如何发送单聊、多聊信息，如何接收、解析微信消息，了解如何实现企业 IM 与微信的交流沟通。

本章主要涉及的知识点有：

- 什么是企业会话模式：学习企业会话模式基础知识。
- 会话聊天：学习单聊、多聊的消息格式，学会如何发送单聊、多聊信息。
- 回调消息：学会如何接收、解析微信会话聊天消息。
- 消息免打扰：了解消息免打扰接口的格式及使用。
- 业务与接口的实际应用：通过本章最后的示例，学习企业 IM 与微信接口的开发，熟练掌握微信企业会话聊天。

6.1 企业会话模式介绍

企业会话模式（以下简称“企业会话”）开发相当于简易版的企业号开发，它具有单独的 AccessToken、Secret，以及单独的回调链接，同样需要对主动推送消息的 AccessToken 进行缓存处理，也同样需要对接收到的消息进行解密、解析、响应，实现消息的双向互动。以企业 IM 接收微信消息流程为例，流程图如图 6.1 所示。企业会话就像企业号的“小号”，加密、缓存规则都是一致的，所以本章不会详细介绍缓存、加密等部分，大家可以通过第 3 章、第 4 章学习。

企业会话的优势在于，可以减少企业 IM 的 App 开发成本，通过企业会话服务，不仅企业号成员可基于微企通讯录直接发起微信对话，而且读者还可以通过接口实现与企业 IM 系统的对接，实现双向同步会话，如图 6.2 所示。

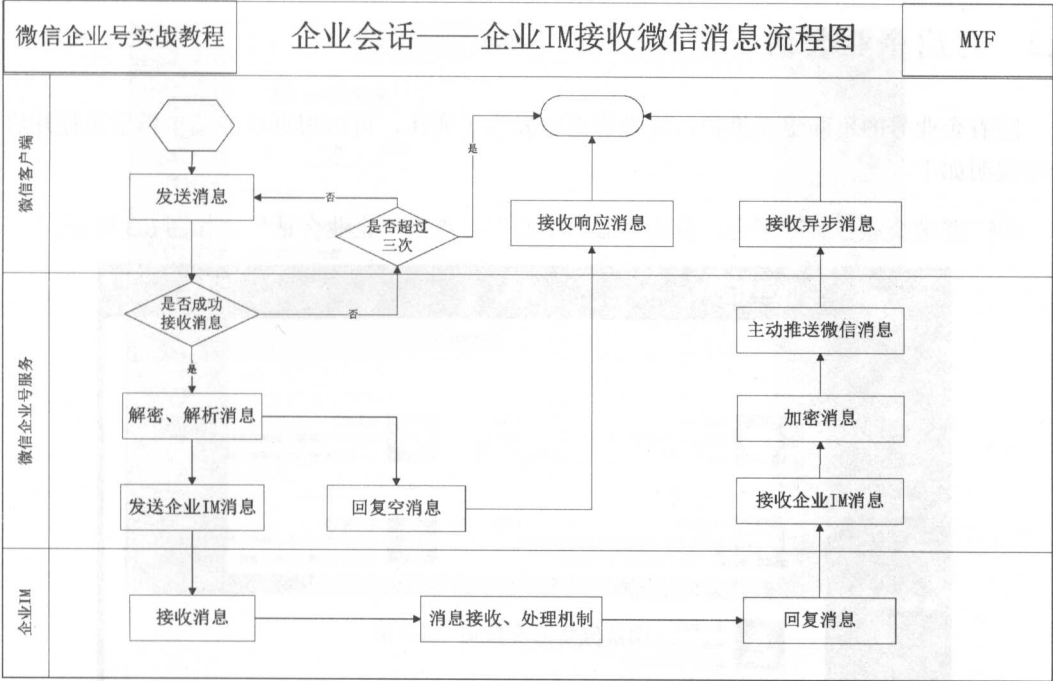


图 6.1 企业 IM 接收微信消息流程图



图 6.2 查找“企业会话”

对于企业号的配置，也需要注意以下几点：

- 企业 IM 中的组织架构需要与企业号中的一致，可以通过异步任务进行定期同步（通讯录异步任务将在第 7 章介绍）。
- 以超级管理员身份开通“企业会话”服务，并开启回调接口（企业会话的开启在 6.2 节中介绍），获得 CorpID 和 Secret。获得的管理组 Secret 为新的接口权限票据，仅支持企业会话内消息互动以及素材管理，不支持应用中的菜单管理权限等。
- 注意配置拥有会话发起人。
- 与被动回调模式一样，企业会话的回调链接需要完成消息响应。对于推送失败的消息，企业将重新推送，推送最大误差为一天。

6.2 开启企业会话

随着企业号的不断更新维护，管理界面也发生了变化，可以根据功能菜单指示进行开启，开始说明如下。

01 登录企业号管理平台，点击“服务中心”，点击“企业会话”，如图 6.3 所示。

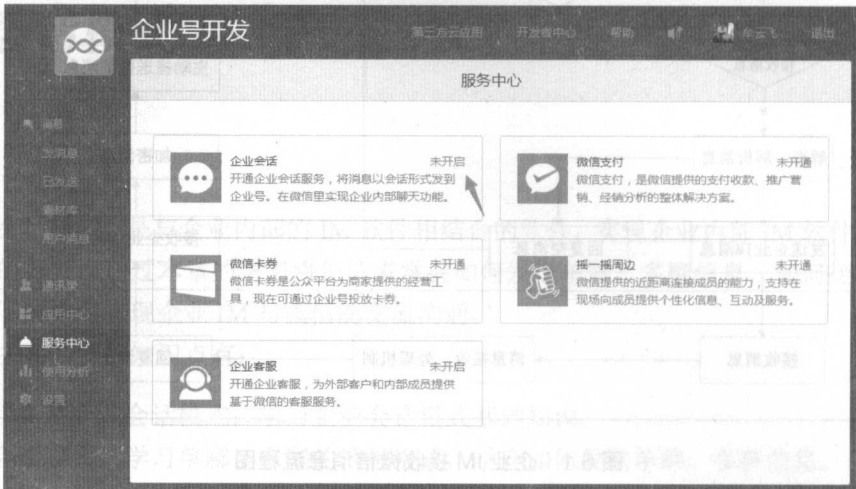


图 6.3 查找“企业会话”

02 进入“企业会话启用”页面，单击“开始使用”，选择“使用范围”之后，单击“更多设置”，如图 6.4 所示。



图 6.4 企业会话设置页面

03 单击发起会话权限中的“添加”，设置允许发起会话的人员名单，可以根据部门、标签、成员进行选择，如果是所有人，则单击部门的根节点（即全部部门），如图 6.5 所示。

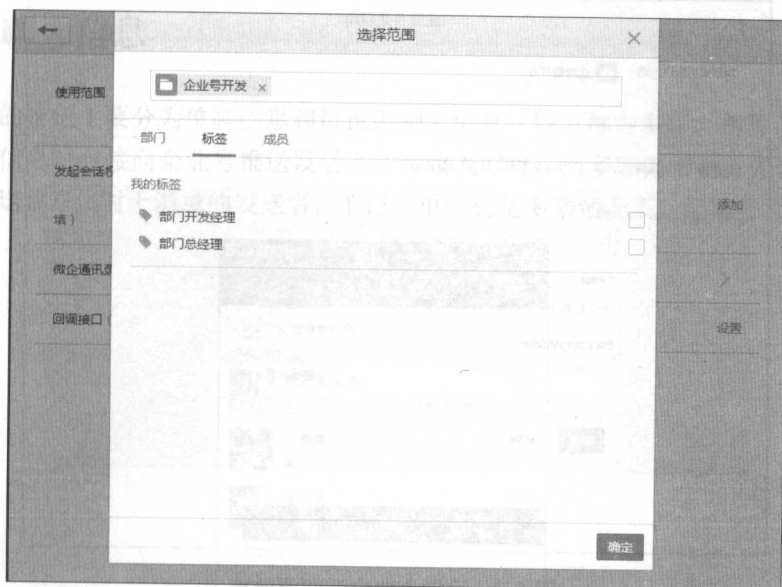


图 6.5 设置会话发起范围

04 设置微企通讯录中成员发起权限，如图 6.6 所示。

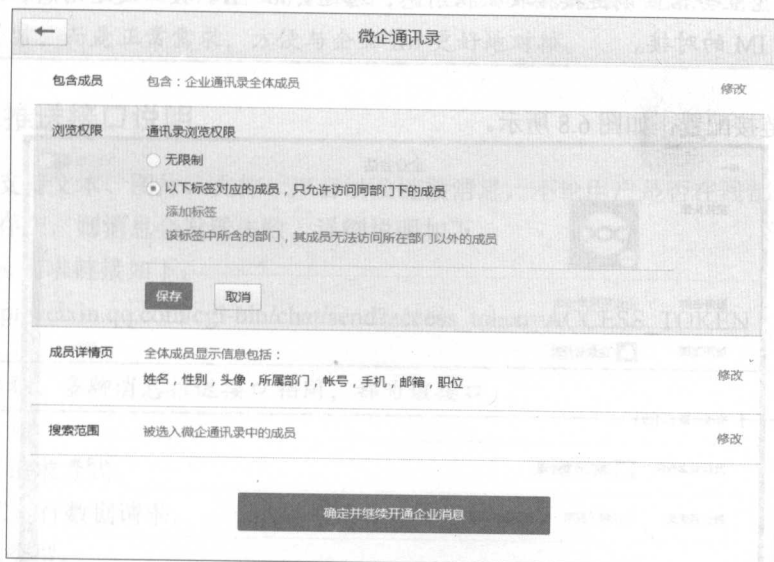


图 6.6 微企通讯录发起权限

备注：微企通讯录为企业号创建时默认创建的应用，读者可以关闭默认应用，创建自己的通讯录应用。

05 设置回调链接，设置回调链接之后，企业就能够接收微信会话消息，实现微信会话的同步，如图 6.7 所示。

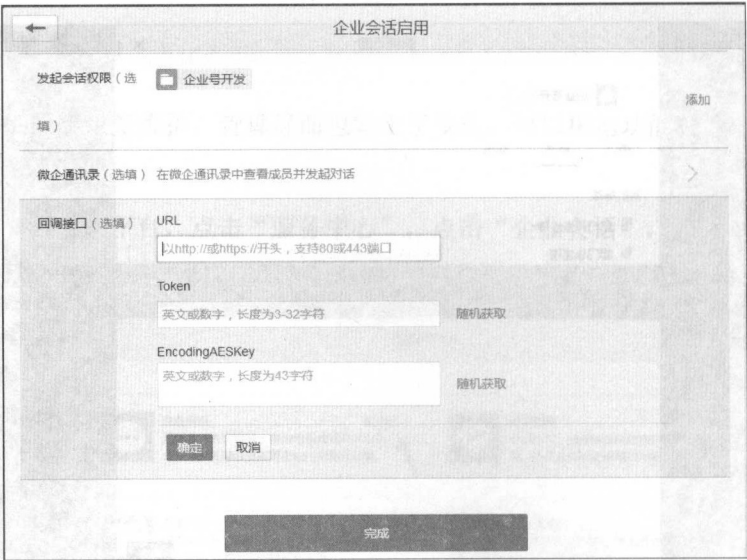


图 6.7 设置企业会话回调链接



备注：企业会话回调开启模式与“第 4 章 被动回调模式”开启方式一致，均需要 GET 验证。通过企业会话回调链接接收微信消息，通过会话 API 接口发送消息，则可以实现微信与企业 IM 的对接。

06 完成链接配置，如图 6.8 所示。



图 6.8 企业会话详细页面

6.3 推送聊天信息

聊天消息的推送主要分为单聊信息和讨论组聊天信息（以下称为多聊）两种，如图 6.9 所示。推送单聊信息时直接向企业号推送发送人和接收人即可，而多聊则需要首先创建讨论组，然后才能够发送信息。对于消息的发送者，不论在单聊还是多聊情况下，都不会收到消息提醒。



图 6.9 企业会话微信客户端界面

注意：在通讯录中但未关注企业号的人员，也能够加入讨论组，也能够对其发送信息，这并不是异常，而是正常需求，方便与企业 IM 更好地对接。

6.3.1 信息推送接口说明

消息发送支持文本、图片、文件、声音以及链接消息，不论用户是否在线都将推送消息。如果接收人不存在，则消息会发送失败，详细说明如下。

(1) Https 请求链接如下：

```
https://qyapi.weixin.qq.com/cgi-bin/chat/send?access_token=ACCESS_TOKEN
```

备注：单聊、多聊消息推送接口相同，都为该接口。

(2) 数据请求方式。

POST 方式进行数据请求。

(3) 消息类型。

支持文本、图片、语音、文件、链接消息。

(4) 权限设置。

发送者需要具有会话发起权限；接收人需要全部存在。

注意：企业 IM 端发送的消息，在同步到发送者的微信上时，不会有提醒，除发送者之外的其他接收人员，在微信客户端将收到提醒。

(5) 示例代码：


```

/**
 * 企业 IM 端发送的消息，在同步到发送者的微信上时，不会有提醒
 * 除发送者之外的其他接收人员，将收到提醒
 * @param tagId 标签 id
 * @return
 */
public JSONObject send_M_Message(String jsonMessage) {
    boolean flag=true;
    //获取微信号
    String token=getTalkTokenFromWx();
    try {
        CloseableHttpClient httpclient = HttpClients.createDefault();
        HttpPost httpPost= new HttpPost("https://qyapi.weixin.qq.com/cgi-bin/chat/send?access_token="+token);
        //发送 JSON 格式的数据
        StringEntity myEntity = new StringEntity(jsonMessage,
            ContentType.create("text/plain", "UTF-8"));
        httpPost.setEntity(myEntity);
        ResponseHandler<JSONObject> responseHandler = new
ResponseHandler<JSONObject>() {
            public JSONObject handleResponse( final HttpResponse response)
throws ClientProtocolException, IOException {
                int status = response.getStatusLine().getStatusCode();
                if (status >= 200 && status < 300) {
                    HttpEntity entity = response.getEntity();
                    if(null!=entity){
                        String result= EntityUtils.toString(entity);
                        //根据字符串生成 JSON 对象
                        JSONObject resultObj = JSONObject.fromObject
(result);
                        return resultObj;
                    }else{
                        return null;
                    }
                } else {
                    throw new ClientProtocolException("Unexpected status:
" + status);
                }
            }
        };
        //返回的 JSON 对象
        JSONObject responseBody = httpclient.execute(httpPost,
responseHandler);
        System.out.println(responseBody);
        return responseBody;
    } catch (Exception e) {

```

```
e.printStackTrace();
return null;
}
}
```

备注：getTalkTokenFromWx 方法为企业会话下的 AccessToken 缓存处理方法，处理方式与主动调用模式下的 AccessToken 一致（3.3 AccessToken 的缓存处理），读者可参照 3.3 节新增两个变量，处理企业会话下的 AccessToken 即可。

6.3.2 聊天消息体结构说明

在开发过程中，需要注意消息体的参数值，除了消息类型不一致导致的参数不同之外，还需要注意单聊、多聊不同场景下的参数值问题。例如，单聊消息 type 值为“single”，多聊消息 type 值为“group”；单聊中 ID 值为接收人企业号中的 ID（即 userid），而多聊中的 ID 为讨论组 ID（讨论组的维护将在 6.3.3 节介绍）。

(1) Text（文字消息）。

文字聊天消息推送格式如下：

```
{
  "receiver":
  {
    "type": "single",
    "id": "lisi"
  },
  "sender": "zhangsan",
  "msgtype": "text",
  "text":
  {
    "content": "Holiday Request For Pony(http://xxxxx)"
  }
}
```

单聊、多聊 text 消息体结构详细说明如表 6.1 所示。

表 6.1 单聊、多聊text消息体结构说明

参数	是否必需	说明
receiver	是	接收人
type	是	接收人类型：single group，分别表示：单聊 多聊
id	是	接收人的值，为userid chatid，分别表示：成员ID 会话ID
sender	是	发送人
msgtype	是	消息类型，此时固定为：text
content	是	消息内容

(2) image（图片消息）。

图片聊天消息推送格式如下：

```
{
  "receiver":
  {
    "type": "group",
    "id": "235364212115767297"
  },
  "sender": "zhangsan",
  "msgtype": "image",
  "image":
  {
    "media_id": "MEDIA_ID"
  }
}
```

备注：图片信息推送的是 media_id，图片信息需要优先上传至微信服务器。由于永久素材有数量限制，因此建议上传临时图片素材。

单聊、多聊 image 消息体结构详细说明如表 6.2 所示。

表 6.2 单聊、多聊image消息体结构说明

参数	是否必需	说明
receiver	是	接收人
type	是	接收人类型：single group，分别表示：单聊 多聊
id	是	接收人的值，为userid chatid，分别表示：成员ID 会话ID
sender	是	发送人
msgtype	是	消息类型，此时固定为：image
media_id	是	图片media_id，可以调用上传素材文件接口获取

(3) file（文件消息）。

文件聊天消息推送格式如下：

```
{
  "receiver":
  {
    "type": "group",
    "id": "235364212115767297"
  },
  "sender": "zhangsan",
  "msgtype": "file",
  "file":
  {
    "media_id": "MEDIA_ID"
  }
}
```



备注：文件信息与图片消息基本相同，建议上传临时文件素材。

单聊、多聊 file 消息体结构详细说明如表 6.3 所示。

表 6.3 单聊、多聊file消息体结构说明

参数	是否必需	说明
receiver	是	接收人
type	是	接收人类型：single group，分别表示：单聊 多聊
id	是	接收人的值，为userid chatid，分别表示：成员ID 会话ID
sender	是	发送人
msgtype	是	消息类型，此时固定为：file
media_id	是	图片media_id，可以调用上传素材文件接口获取，文件必须大于4字节

(4) voice（音频消息）。

音频聊天消息推送格式如下：

```
{
  "receiver":
  {
    "type": "group",
    "id": "235364212115767297"
  },
  "sender": "zhangsan",
  "msgtype": "voice",
  "voice":
  {
    "media_id": "MEDIA_ID"
  }
}
```



备注：音频信息与图片消息基本相同，建议上传临时文件素材。

单聊、多聊 voice 消息体结构详细说明如表 6.4 所示。


表 6.4 单聊、多聊voice消息体结构说明

参数	是否必需	说明
receiver	是	接收人
type	是	接收人类型：single group，分别表示：单聊 多聊
id	是	接收人的值，为userid chatid，分别表示：成员ID 会话ID
sender	是	发送人
msgtype	是	消息类型，此时固定为：voice
media_id	是	图片media_id，可以调用上传素材文件接口获取，文件大小必须大于4字节

(5) link（链接消息）

链接聊天消息推送格式如下：

```
{
  "receiver":
  {
    "type": "single",
    "id": "lisi"
  },
  "sender": "zhangsan",
  "msgtype": "link",
  "link":
  {
    "title": "CSDN 博客",
    "description": "link 链接消息描述",
    "url": " http://blog.csdn.net/myfmyfmyfmyf ",
    "thumb_media_id":
    "177fIcVBfOLYa703hzBByU1EH3_sdp4hy yaxN4Gfdc-o66vG7k-lXgEacQqfuCcJ-VbZnPlUKJD
F8ig_8Zgh6-g"
  }
}
```

 **备注：**需要注意 title、description 文字勿超过限制数。

单聊、多聊 link 消息体结构详细说明如表 6.5 所示。

表 6.5 单聊、多聊link消息体结构说明

参数	是否必须	说明
receiver	是	接收人
type	是	接收人类型：single group，分别表示：单聊 多聊
id	是	接收人的值，为userid chatid，分别表示：成员ID 会话ID
sender	是	发送人
msgtype	是	消息类型，此时固定为：link
title	是	消息标题，不超过128字节
description	否	消息描述，不超过512字节
url	是	跳转的URL
thumb_media_id	是	图片media_id，可以调用上传素材文件接口获取

6.3.3 创建多聊会话

单人聊天只需向指定的单个人员发送信息，而多人聊天则需要先将信息人“圈”起来，将大家聚集在一起然后发送信息。本节将为读者介绍如何将信息接收人聚集起来，并建立会话讨论组。

(1) Https 请求链接如下：

https://qyapi.weixin.qq.com/cgi-bin/chat/create?access_token=ACCESS_TOKEN

(2) 数据请求方式。

POST 方式进行数据请求。

(3) 消息请求结构体。

此方式是通过接口创建讨论组，如果通过微信客户端创建讨论组，那么读者服务端将收到微信创建讨论组的回调信息，消息请求结构体如下：

```
{
  "chatid": "1",
  "name": "企业应用中心",
  "owner": "zhangsan",
  "userlist": ["zhangsan", "lisi", "wangwu"]
}
```

注意：userlist 中包含 owner（会话创建人）；发起人必须具有发起会话权限；会话中人员必须存在。

(4) 请求参数说明。详细说明如表 6.6 所示。

表 6.6 创建讨论组请求参数说明

参数	是否必需	说明
chatid	是	会话ID。字符串类型，最长32个字符。只允许字符0-9及字母a-z、A-Z。如果值内容为64bit无符号整型：要求值范围在[1, 2^63)之间，[2^63, 2^64)为系统分配会话id区间
name	是	会话标题
owner	是	管理员UserID，必须是该会话UserList的成员之一
userlist	是	会话成员列表，成员用UserID来标识。会话成员必须在3人或以上，1000人以下

使用技巧：对于企业 IM 中存在、但微信企业号中不存在的用户，且只有三个人的会话，可以通过增加“企业号小助手”（此人员为读者虚拟出的、具有微信号的人员）来完成企业会话的创建。

(5) 示例代码。

封装聊天会话处理方法 sendMsg2WX，通过会话创建传递处理链接以及 JSON 数据，示例代码如下：

```
/**
 * 向微信传递 JSON 信息，并返回信息
 * @param url 如："https://qyapi.weixin.qq.com/cgi-bin/chat/update?
access_token="
 * @param jsonStr
 * @return
```

```
    */
    public JSONObject sendMsg2WX(String url,String jsonStr){
        //获取微信号
        String token=getTalkTokenFromWx();
        try {
            CloseableHttpClient httpclient = HttpClients.createDefault();
            HttpPost httpPost= new HttpPost(url+token);
            //发送 JSON 格式的数据
            StringEntity myEntity = new StringEntity(jsonStr,ContentType.
create("text/plain", "UTF-8"));
            httpPost.setEntity(myEntity);
            ResponseHandler<JSONObject> responseHandler = new ResponseHandler
<JSONObject>() {
                public JSONObject handleResponse(
                    final HttpResponse response) throws ClientProtocolException,
IOException {
                    int status = response.getStatusLine().getStatusCode();
                    if (status >= 200 && status < 300) {
                        HttpEntity entity = response.getEntity();
                        if(null!=entity){
                            String result= EntityUtils.toString(entity);
                            //根据字符串生成 JSON 对象
                            JSONObject resultObj = JSONObject.fromObject
(result);
                            return resultObj;
                        }else{
                            return null;
                        }
                    } else {
                        throw new ClientProtocolException("Unexpected response
status: " + status);
                    }
                }
            };
            //返回的 JSON 对象
            JSONObject responseBody = httpclient.execute(httpPost, responseHandler);
            System.out.println(responseBody);
            return responseBody;
        }catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

封装会话聊天处理方法:

```
/**
```

```

* 创建微信聊天组 JSON
* @param util 工具类
* @param wxChatId 会话 Id
* @param title 会话标题
* @param owner 会话发起人
* @param userList 会话人员列表
* @param assistIsExit 是否需要小助手
* @return 返回创建信息
*/

private JSONObject createChatWindow(WxUtil util,String wxChatId,String
title,String owner,String userList,boolean assistIsExit){
    //是否需要增加"企业号小助手"
    if(true==assistIsExit){
        userList+=",\""+ WxUtil.ASSIST_CODE+"\"";
    }
    //创建会话 JSON 数据
    String jsonStr="{\"chatid\": \""+wxChatId+"\", \"name\": \""+title+
    "\", \"owner\": \""+owner+"\", \"userlist\": \""+userList+"\"}";
    JSONObject jsobObj= util.sendMsg2WX(WxUtil.CREATE_CHAT_URL, jsonStr);
    return jsobObj;
}

```



备注：企业号小助手（WxUtil.ASSIST_CODE）既可以使用管理员的微信号代替，也可以虚拟一个成员。

(6) 返回结果。

返回结果主要以 `errcode` 参数为主，0 代表成功，1 代表失败，示例结果如下所示：

```

{
  "errcode": 0,
  "errmsg": "ok"
}

```

6.3.4 修改多聊会话

讨论组可能出现人员增加、修改标题等操作，这就需要会话的修改操作，接口说明如下。

(1) Https 请求链接如下：

https://qyapi.weixin.qq.com/cgi-bin/chat/update?access_token=ACCESS_TOKEN

(2) 数据请求方式。

POST 方式进行数据请求。

(3) 消息请求结构体。

```

{
  "chatid": "296875212115767297",
  "op_user": "muyunfei",

```




```
    "name": "企业应用中心",
    "owner": "zhangsan",
    "add_user_list": ["lisi"],
    "del_user_list": ["wangwu"]
}
```

(4) 请求参数说明。详细说明如表 6.7 所示。

表 6.7 修改讨论组请求参数说明

参数	是否必需	说明
chatid	是	会话ID
op_user	是	操作人UserID
name	否	会话标题
owner	否	管理员UserID必须是该会话userlist的成员之一
add_user_list	否	会话新增成员列表，成员用UserID来标识
del_user_list	否	会话退出成员列表，成员用UserID来标识

 备注：chatid 为已经创建的讨论组 ID，不能是读者新生成的 ID。


(5) 示例代码：

```
/**
 * 修改会话信息
 * @param util 工具类
 * @param addList 添加的人员名单
 * @param deleteList 删除的人员名单
 * @param name 会话标题
 * @param owner 拥有者
 * @param oper 操作者
 * @param wxChatId 讨论组 ID
 * @return
 */
private JSONObject updateChatWindow(WxUtil util, List<WxChatMessageGroup>
addList, List<WxChatMessageGroup> deleteList, String name, String owner, String
oper, String wxChatId) {
    String jsonStr="{\"chatid\": \""+wxChatId+"\","+
        +\"\"op_user\": \""+owner+"\","+
        +\"\"name\": \""+name+"\","+
        +\"\"owner\": \""+owner+"\","+
        String addStr="";
    //将 list 转变成字符串
    if(null!=addList&&0!=addList.size()){
        for (int i = 0; i < addList.size(); i++) {
            if(null!=addStr&&"".equals(addStr)){
                addStr+=",";
            }
        }
    }
}
```

```

        addStr=addStr+"\""+addList.get(i).getUserCode()+"\"";
    }
    jsonStr+="\"add_user_list\":["+addStr+"]";
}
String deleteStr="";
if(null!=deleteList&&0!=deleteList.size()){
    for (int i = 0; i < deleteList.size(); i++) {
        if(null!=deleteStr&&"".equals(deleteStr)){
            deleteStr+=",";
        }
        deleteStr=deleteStr+"\""+deleteList.get(i).getUserCode()+"\"";
    }
    jsonStr+="\"del_user_list\":["+deleteStr+"]";
}
//发送修改会话请求,并获得结果
JSONObject jsobObj=util.sendMsg2WX(WxUtil.UPDATE_CHAT_URL, jsonStr);
return jsobObj;
}

```

 备注: sendMsg2WX 方法为封装方法在 6.3.3 节中介绍。

(6) 返回结果。

返回结果主要以 errcode 参数为主, 0 代表成功, 1 代表失败, 示例结果如下所示:

```

{
    "errcode": 0,
    "errmsg": "ok"
}

```

6.3.5 退出多聊会话

成员可以通过企业 IM 退出讨论组, 这时需要对应的微信客户端也作出相应的响应, 退出多聊会话组, 此操作需要通过接口实现, 接口说明如下。

(1) Https 请求链接如下:

https://qyapi.weixin.qq.com/cgi-bin/chat/quit?access_token=ACCESS_TOKEN

(2) 数据请求方式。

POST 方式进行数据请求。

(3) 消息请求结构体:

```

{
    "chatid": "296875212115767297",
    "op_user": "muyunfei",
}

```


(4) 请求参数说明。详细说明如表 6.8 所示。

表 6.8 退出讨论组请求参数说明

参数	是否必需	说明
chatid	是	会话ID
op_user	是	操作人UserID

(5) 示例代码:

```
//退出会话
String jsonStr3="{\"chatid\": \"wx12345678\", \"op_user\": \"wangjiankun\"}";
demo.sendMsg2WX(URL_EXIT_CHAT_GROUP, jsonStr3);
```

 备注: sendMsg2WX 方法为封装方法, 在 6.3.3 节中曾介绍过。URL_EXIT_CHAT_GROUP 为请求链接全局变量。

(6) 返回结果。

返回结果主要以 errcode 参数为主, 0 代表成功, 1 代表失败, 示例结果如下所示:

```
{
  "errcode": 0,
  "errmsg": "ok"
}
```

6.3.6 获取多聊会话信息

获取企业会话讨论组详细信息, 接口说明如下。

(1) Https 请求链接如下:

https://qyapi.weixin.qq.com/cgi-bin/chat/get?access_token=ACCESS_TOKEN&chatid=CHATID

(2) 数据请求方式。

GET 方式进行数据请求。

(3) 请求参数说明。详细说明如表 6.9 所示。


表 6.9 获取讨论组信息请求参数说明

参数	是否必需	说明
Access_token	是	企业会话接口票据
chatid	是	讨论组ID

(4) 示例代码:

```
//获得会话信息
String url="https://qyapi.weixin.qq.com/cgi-bin/chat/get?chatid=
wx12345678&access_token=";
String jsonStr4="";
JSONObject result = demo.sendMsg2WX(url, jsonStr4);
System.out.println(result.toString());
//获得 chatid
```

```
System.out.println(result.getJSONObject("chat_info").get("chatid"));
//获得 userlist 中第一条数据
System.out.println(result.getJSONObject("chat_info").getJSONArray("userl
ist").get(0));
```

 **备注：**sendMsg2WX 方法为封装方法，在 6.3.3 节中曾介绍过。获得结果为 JSONObject 对象（net.sf.json.JSONObject），对于 chat_info 对象，可以使用 getJSONObject("chat_info") 方法获得；对于 userlist 数组，可以通过 getJSONArray("userlist") 获得。

(5) 返回结果。

返回结果主要以 errcode 参数为主，0 代表成功，1 代表失败，示例结果如下所示：

```
{
  "errcode": 0,
  "errmsg": "ok",
  "chat_info":
  {
    "chatid": "235364212115767297",
    "name": "企业应用中心",
    "owner": "zhangsan",
    "userlist": ["zhangsan", "lisi", "wangwu"]
  }
}
```

返回结果详细说明如表 6.10 所示。

表 6.10 讨论组获取信息参数说明

参数	说明
errcode	返回码
errmsg	返回码的文本描述信息
chat_info	会话信息
chatid	会话ID
name	会话标题
owner	管理员UserID
userlist	会话成员列表，成员用UserID来标识

6.3.7 清除未读会话状态

标识会话状态为已读，消除未读会话状态，接口说明如下。

(1) Https 请求链接如下：

https://qyapi.weixin.qq.com/cgi-bin/chat/clearnotify?access_token=ACCESS_TOKEN

(2) 数据请求方式。

POST 方式进行数据请求。


(3) 消息请求结构体：


```
{
  "op_user": "zhangsan",
  "chat": {
    "type": "single",
    "id": "lisi"
  }
}
```

(4) 请求参数说明。详细说明如表 6.11 所示。


表 6.11 清除未读会话状态请求参数说明

参数	是否必需	说明
op_user	是	会话所有者的UserID
chat	是	会话
type	是	会话类型：single group，分别表示：单聊 多聊
id	是	会话值，为userid chatid，分别表示：成员ID 会话ID

 备注：当 type 值为 single 时，id 值为成员 id (userid)；当 type 值为 group 时，id 值为会话 id (chatid)。

(5) 示例代码：

```
//清除未读状态
String jsonStr3="{\"op_user\": \"muyunfei\", \"chat\": {\"type\": \"group\", \"id\": \"wx12345678\"}}";
demo.sendMsg2WX(URL_CLEAR_CHAT_GROUP, jsonStr3);
```

 备注：sendMsg2WX 方法为封装方法，在 6.3.3 节中曾介绍过。URL_CLEAR_CHAT_GROUP 全局变量为清除未读信息的请求链接。

(6) 返回结果。

返回结果主要以 errcode 参数为主，0 代表成功，1 代表失败，示例结果如下所示：

```
{
  "errcode": 0,
  "errmsg": "ok"
}
```

6.3.8 会话消息免打扰

该接口主要用于设置成员微信客户端是否接收消息提醒，接口说明如下。

(1) Https 请求链接如下：

https://qyapi.weixin.qq.com/cgi-bin/chat/setmute?access_token=ACCESS_TOKEN

(2) 数据请求方式。

POST 方式进行数据请求。

(3) 消息请求结构体:

```
{
  "user_mute_list":
  [
    {
      "userid": "zhangsan",
      "status": 0
    },
    {
      "userid": "lisi",
      "status": 1
    }
  ]
}
```



备注: 该接口可用于以下场景。

- a. 成员在企业 IM 设置微信“消息免打扰”。
- b. 当成员企业 IM 在线时, 微信客户端可以设置“消息免打扰”(也可以不向微信客户端发送信息, 由读者制定规则)。当成员企业 IM 离线或者退出时, 可以关闭“消息免打扰”, 进行微信客户端提醒。

(4) 请求参数说明。详细说明如表 6.12 所示。

表 6.12 清除未读会话状态请求参数说明

参数	是否必需	说明
user_mute_list	是	免打扰成员数组, 最大支持10000个成员
userid	是	成员UserID
status	是	免打扰状态: 0为关闭, 1为打开, 默认为0。当打开时所有消息不提醒; 当关闭时, 以成员对会话的设置为准, 如图6.10所示



备注: 消息免打扰接口, 可一次性提交不超过 10000 人。

(5) 示例代码:

```
//消息免打扰
String jsonStr6="{\"user_mute_list\":["
    +\"{\\"userid\": \"muyunfei\", \"status\": 1},\"
    +\"{\\"userid\": \"yuchunyang\", \"status\": 0}\"
    +\"]}\";
demo.sendMsg2WX(URL_ALERT_CHAT_GROUP, jsonStr6);
```

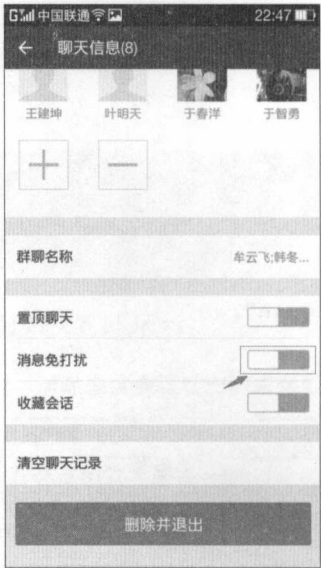


图 6.10 成员会话设置



备注：sendMsg2WX 方法为封装方法，在 6.3.3 节中曾介绍过。返回结果中，invaliduser 参数为无效成员，其余为有效成员，正常执行消息免打扰操作。

(6) 返回结果。

返回结果主要以 errcode 参数为主，0 代表成功，1 代表失败，示例结果如下所示：

```
{
  "errcode": 0,
  "errmsg": "ok",
  "invaliduser": ["zhangsan"]
}
```

6.4 接收聊天信息

消息有发送就有接收，6.3 节我们学习了会话消息的推送，本节将介绍微信会话聊天信息的接收。

6.4.1 信息接收接口说明

接收微信聊天信息，与被动回调模式基本一致，当成员在微信端发消息或更改会话成员时，企业号会推送相应的消息、事件，获得消息之后需要经过 URL 验证、消息解密、消息响应，不同之处在于，回复消息时为明文消息并不是加密消息，而且接收到的数据消息包含多条消息或事件。对于信息的接收需要注意以下几点：

- 当连接失败、请求超时等回调失败时，最大重试间隔为 20 分钟，最大重试时长为 1 天。

- 回复响应消息时，需要明文回复 PackageId 节点值。
- 对于每条消息的排重，可以通过 Item 节点中的 MsgId 或者 FromUserName+ CreateTime 的方式排重。
- 回调数据包中，每条消息为数据包中的一个 Item 节点，数据包包含多条普通消息或事件消息（即多个 Item 节点）。
- 聊天消息的接收与被动回调模式 URL 验证、解密方式一致，读者可以参照第4章学习，接收到的会话信息如下：

```
<xml>
  <Encrypt><![CDATA[ENCRYPT_DATA]]></Encrypt>
  <ToUserName>CORPID</ToUserName>
  <AgentType>chat</AgentType>
</xml>
```

ENCRYPT_DATA 为密文消息，需要解密，解密后的数据包如下所示：

```
<xml>
  <AgentType>chat</AgentType>
  <ToUserName>CORPID</ToUserName>
  <ItemCount>2</ItemCount>
  <PackageId>3156175696255</PackageId>
  <Item>
    <FromUserName><![CDATA[fromUser]]></FromUserName>
    <CreateTime>1348831860</CreateTime>
    <MsgType><![CDATA[event]]></MsgType>
    <Event><![CDATA[update_chat]]></Event>
    <Name><![CDATA[事件消息更新会话]]></Name>
    <Owner><![CDATA[zhangsan]]></Owner>
    <AddUserList><![CDATA[zhaoliu]]></AddUserList>
    <DelUserList><![CDATA[lisi|wangwu]]></DelUserList>
    <ChatId><![CDATA[235364212115767297]]></ChatId>
  </Item>
  <Item>
    <FromUserName><![CDATA[fromUser]]></FromUserName>
    <CreateTime>1348831860</CreateTime>
    <MsgType><![CDATA[event]]></MsgType>
    <Event><![CDATA[事件消息创建会话]]></Event>
    <ChatInfo>
      <ChatId><![CDATA[235364212115767297]]></ChatId>
      <Name><![CDATA[企业应用中心]]></Name>
      <Owner>zhangsan</Owner>
      <UserList>zhangsan|lisi|wangwu</UserList>
    </ChatInfo>
  </Item>
</xml>
```


6.4.2 普通消息结构体说明


普通消息分为六类，包括 text 消息、image 消息、voice 消息、file 消息、link 消息和 location 消息，每类消息又分为单聊消息和多聊消息。对于单聊、多聊信息，只需注意 Receiver 节点即可，单聊信息 Receiver 节点中的 Type 为 single、ID 为成员 userid，而多聊信息 Receiver 节点中的 Type 为 group、ID 为讨论组 id，消息的结构体详细说明如下。

(1) text 文字消息（结构体详细说明如表 6.13 所示）：

```
<Item>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1348831860</CreateTime>
  <MsgType><![CDATA[text]]></MsgType>
  <Content><![CDATA[this is a test]]></Content>
  <MsgId>1234567890123456</MsgId>
  <Receiver>
    <Type>single</Type>
    <Id>lisi</Id>
  </Receiver>
</Item>
```

表 6.13 企业会话text回调消息参数说明

参数	说明
FromUserName	发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：text
Content	消息内容，支持表情，表情见附录A
MsgId	消息ID，64位整型
Receiver	接收人信息
Type	接收人类型：single group，分别表示：单聊 多聊
Id	接收人的值，为userid chatid，分别表示：成员ID 会话ID

 备注：普通消息可以直接使用 MsgId 进行排重。


(2) image 图片消息（结构体详细说明如表 6.14 所示）：

```
<Item>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1348831860</CreateTime>
  <MsgType><![CDATA[image]]></MsgType>
  <PicUrl><![CDATA[this is a url]]></PicUrl>
  <MediaId><![CDATA[media_id]]></MediaId>
  <MsgId>1234567890123456</MsgId>
  <Receiver>
    <Type>single</Type>
    <Id>lisi</Id>
```

```
</Receiver>
</Item>
```

表 6.14 企业会话image回调消息参数说明

参数	说明
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：image
PicUrl	图片链接
MediaId	图片media_id，可以调用获取素材文件接口拉取数据
MsgId	消息ID，64位整型
Receiver	接收人信息
Type	接收人类型：single group，分别表示：单聊 多聊
Id	接收人的值，为userid chatid，分别表示：成员ID 会话ID

 **备注：**通过 MediaId 获得图片、音频、文件等素材，读者可以通过 3.6 节学习，本节不再介绍。

(3) voice 音频消息（结构体详细说明如表 6.15 所示）：

```
<Item>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1348831860</CreateTime>
  <MsgType><![CDATA[voice]]></MsgType>
  <MediaId><![CDATA[media_id]]></MediaId>
  <MsgId>1234567890123456</MsgId>
  <Receiver>
    <Type>single</Type>
    <Id>lisi</Id>
  </Receiver>
</Item>
```

表 6.15 企业会话voice回调消息参数说明


参数	说明
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：voice
MediaId	语音media_id，可以调用获取素材文件接口拉取数据
MsgId	消息ID，64位整型
Receiver	接收人信息
Type	接收人类型：single group，分别表示：单聊 多聊
Id	接收人的值，为userid chatid，分别表示：用户ID 会话ID

(4) file 文件消息（结构体详细说明如表 6.16 所示）：

```
<Item>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1348831860</CreateTime>
  <MsgType><![CDATA[file]]></MsgType>
  <MediaId><![CDATA[media_id]]></MediaId>
  <MsgId>1234567890123456</MsgId>
  <Receiver>
    <Type>single</Type>
    <Id>lisi</Id>
  </Receiver>
</Item>
```

表 6.16 企业会话file回调消息参数说明

参数	说明
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：file
MediaId	语音media_id，可以调用获取素材文件接口拉取数据
MsgId	消息ID，64位整型
Receiver	接收人信息
Type	接收人类型：single group，分别表示：单聊 多聊
Id	接收人的值，为userid chatid，分别表示：用户ID 会话ID

 **备注：**file 消息与 link 消息成员可以通过转发或者我的收藏发送消息。

(5) link 链接消息（结构体详细说明如表 6.17 所示）：


```
<Item>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1348831860</CreateTime>
  <MsgType><![CDATA[link]]></MsgType>
  <Title><![CDATA[TITLE]]></Title>
  <Description><![CDATA[DESCRIPTION]]></Description>
  <Url><![CDATA[URL]]></Url>
  <PicUrl><![CDATA[PIC_URL]]></PicUrl>
  <MsgId>1234567890123456</MsgId>
  <Receiver>
    <Type>single</Type>
    <Id>lisi</Id>
  </Receiver>
</Item>
```

表 6.17 企业会话link回调消息参数说明

参数	说明
FromUserName	信息发送人，成员UserID

续表

参数	说明
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：link
Title	标题
Description	描述
Url	链接
PicUrl	图片链接
MsgId	消息ID，64位整型
Receiver	接收人信息
Type	接收人类型：single group，分别表示：单聊 多聊
Id	接收人的值，为userid chatid，分别表示：用户id 会话id

 备注：通过 URL 链接利用 httpclient 或者 httpurlconnection 等方式实现数据抓取。

6.4.3 事件消息结构体说明

接收聊天消息的时间与推送聊天事件消息一致，均为多聊会话事件，包括：创建会话事件、修改会话事件和退出会话事件，每条事件消息同样作为回调数据包中的一个 Item 节点，事件详细说明如下。

(1) 创建会话事件（结构体详细说明如表 6.18 所示）：


```
<Item>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1348831860</CreateTime>
  <MsgType><![CDATA[event]]></MsgType>
  <Event><![CDATA[create_chat]]></Event>
  <ChatInfo>
    <ChatId><![CDATA[235364212115767297]]></ChatId>
    <Name><![CDATA[企业应用中心]]></Name>
    <Owner>zhangsan</Owner>
    <UserList>zhangsan|lisi|wangwu</UserList>
  </ChatInfo>
</Item>
```

表 6.18 创建会话事件回调消息参数说明

参数	说明
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：event
Event	事件类型，此时固定为：create_chat
ChatId	会话ID
Name	会话标题

续表

参数	说明
Owner	管理员UserID
UserList	会话成员列表，成员用UserID标识，成员间以竖线“ ”分隔

 **使用技巧：**UserList 中成员 ID 使用“|”分隔，可以使用 `String[] userArray=userStr.split("|");` 方法进行分隔得到 String 数组。

(2) 修改会话事件（结构体详细说明如表 6.19 所示）：

```
<Item>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1348831860</CreateTime>
  <MsgType><![CDATA[event]]></MsgType>
  <Event><![CDATA[update_chat]]></Event>
  <Name><![CDATA[企业应用中心]]></Name>
  <Owner><![CDATA[zhangsan]]></Owner>
  <AddUserList><![CDATA[zhaoliu]]></AddUserList>
  <DelUserList><![CDATA[lisi|wangwu]]></DelUserList>
  <ChatId><![CDATA[235364212115767297]]></ChatId>
</Item>
```

表 6.19 修改会话事件回调消息参数说明

参数	说明
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：event
Event	事件类型，此时固定为：update_chat
Name	会话名称
Owner	会话所有者（管理员）
AddUserList	会话新增成员列表，成员间以竖线分隔，如：zhangsan lisi
DelUserList	会话删除成员列表，成员间以竖线分隔，如：zhangsan lisi
ChatId	会话ID

(3) 退出会话事件（结构体详细说明如表 6.20 所示）：

```
<Item>
  <FromUserName><![CDATA[fromUser]]></FromUserName>
  <CreateTime>1348831860</CreateTime>
  <MsgType><![CDATA[event]]></MsgType>
  <Event><![CDATA[quit_chat]]></Event>
  <ChatId><![CDATA[235364212115767297]]></ChatId>
</Item>
```

表 6.20 退出会话事件回调消息参数说明

参数	说明
FromUserName	信息发送人，成员UserID
CreateTime	消息创建时间（整型）
MsgType	消息类型，此时固定为：event
Event	事件类型，此时固定为：quit_chat
ChatId	会话ID

6.5 案例：企业 IM 与微信的对接

通过对前几节的学习，我们已经了解了什么是企业会话模式，以及企业会话模式中的各个接口，本节将介绍开发企业 IM 与微信对接的中间服务。为什么需要创建中间服务，直接在企业 IM 中集成是否可行？企业 IM 中可以直接集成，但为了增加系统的独立性和容灾性，防止中间服务的异常，而影响整体企业 IM。其次，企业 IM 多以内网服务部署，而微信为外网服务，中间服务的创建多以 DMZ 区（危险管理区，将在第 8 章介绍）为主，所以创建中间服务也是必要的。

微信与企业 IM 互动的中间服务分为两类：一类是转发企业 IM 的消息服务类，这里我们命名为 sendMessageServlet.java；另一类则是转发微信消息的消息服务类，这里我们命名为 ChatMessageServlet.java。两个服务类的详细说明如下。

（1）转发企业 IM 消息服务类 sendMessageServlet.java。

企业 IM 消息服务数据传递以 JSON 格式数据流形式传递，使用 req.getInputStream()获得数据流进行传递：

```
/**
 * 企业 IM，信息服务类
 * 接收企业 IM 信息，并转发至微信客户端
 */
public class sendMessageServlet extends HttpServlet {
    protected void doGet(HttpServletRequest arg0, HttpServletResponse arg1)
        throws ServletException, IOException {
        this.doPost(arg0, arg1);
    }
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        //从输入流读取内容
        BufferedReader br = new BufferedReader(new InputStreamReader
(req.getInputStream()));
        String line = null;
        StringBuilder sb = new StringBuilder();
        while((line = br.readLine())!=null){
            sb.append(line);
        }
    }
}
```

```

String content=sb.toString();
//正则去除\
content=content.replaceAll("\\\\", "\\\\\\\");
//获取调用方法
String action =req.getParameter("action");
String resultMsg="";//返回结果
if("sendMessage".equals(action)){
    //发送单个消息
    resultMsg=sendMessage(content);
}else if("sendGroupMessage".equals(action)){
    //
    resultMsg=sendGroupMessage(content);
}
//返回输出结果
OutputStream out = resp.getOutputStream();
out.write(resultMsg.getBytes());
out.flush();
out.close();
//resp.getOutputStream();
}
}

```



备注：sendMessage 发送单聊信息方法与 sendGroupMessage 发送多聊信息可以参照 6.3.1 节创建。对于特殊字符“\”的转义可以使用 replaceAll("\\\\", "\\\\\\\")。

在 web.xml 中创建 servlet 服务连接，示例代码如下：

```

<!-- 企业 IM 消息发送接口 -->
<servlet>
    <servlet-name>webservice1</servlet-name>
    <servlet-class>
        com.webservice.sendMessageServlet
    </servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>webservice1</servlet-name>
    <url-pattern>/webservice.slt</url-pattern>
</servlet-mapping>

```

测试连接示例代码如下：

```

public class TestMain {
    public static void main(String[] args) {
        //m+单人消息
        //String jsonMessage="{\"mchatid\": \"1\", \"receiver\": \"muyunfei\",
        \"sender\": \"xinyueqiao\", \"msgtype\": \"text\", \"content\": \"特殊符号\\\\\"}";
        //m+讨论组
    }
}

```

```

String jsonMessage = "{\"mchatid\": \"1\", \"
    + \"name\": \"小伙伴\", \"
    + \"owner\": \"muyunfei\", \"
    + \"userlist\": [\"xinyueqiao\", \"muyunfei\",
\"yuyang2\", \"songyanqin\", \"kangxiyao\"], \"
    + \"sender\": \"xinyueqiao\", \"
    + \"msgtype\": \"text\", \"
    + \"content\": \"特殊符号: + - * % ( ) { } !, #, $, &,
', (, ), *, +, ,, -, ., /, :, ;, =, ?, @, _, ~, 0-9, a-z, A-Z, \\ \\r\\n 表情/:, @f/:pd
\\}\"";

try {
    CloseableHttpClient httpClient = HttpClients.createDefault();
    // 单聊
    //HttpPost httpPost= new HttpPost("http://服务地址/demo/
//webservice.slt?action=sendSingleMessage");
    HttpPost httpPost= new HttpPost("http://服务地址/demo/webservice.
slt?action=sendGroupMessage");
    // 发送 JSON 格式的数据
    StringEntity myEntity = new StringEntity(jsonMessage, ContentType.
create("text/json", "GBK"));
    httpPost.setEntity(myEntity);
    ResponseHandler<JSONObject> responseHandler = new ResponseHandler
<JSONObject>() {
        public JSONObject handleResponse(
            final HttpResponse response) throws ClientProtocolException,
IOException {
            int status = response.getStatusLine().getStatusCode();
            if (status >= 200 && status < 300) {
                HttpEntity entity = response.getEntity();
                if (null != entity) {
                    String result = EntityUtils.toString(entity);
                    // 根据字符串生成 JSON 对象
                    JSONObject resultObj = JSONObject.fromObject
(result);

                    return resultObj;
                } else {
                    return null;
                }
            } else {
                throw new ClientProtocolException("Unexpected: " + status);
            }
        }
    };

    JSONObject responseBody = httpClient.execute(httpPost, responseHandler);
    System.out.println(responseBody);
} catch (Exception e) {

```



```

        e.printStackTrace();
    }
}
}

```

(2) 转发微信消息服务类 ChatMessageServlet.java。

doGet 方法用于配置回调链接时的链接有效性验证，doPost 方法用于接收真正的数据包。数据包经过解密之后将获得多条 Item 节点数据，对 Item 解析处理即可，ChatMessageServlet 示例代码如下：

```

/**
 * 企业会话，信息回调服务类
 * 接收微信客户端消息，并转发至企业 IM
 */
public class ChatMessageServlet extends HttpServlet {
    private static final long serialVersionUID = 4440739483644821986L;
    /**
     * 请求校验（确认请求来自微信服务器）
     */
    public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        //微信加密签名
        String signature = request.getParameter("msg_signature");
        System.out.println("signature:"+signature);
        //时间戳
        String timestamp = request.getParameter("timestamp");
        System.out.println("timestamp:"+timestamp);
        //随机数
        String nonce = request.getParameter("nonce");
        System.out.println("nonce:"+nonce);
        //随机字符串
        String echostr = request.getParameter("echostr");
        System.out.println("echostr:"+echostr);
        String sToken = WxUtil.RESP_MESSAGE_TOKEN;
        String sCorpID = WxUtil.RESP_MESSAGE_CORPID;
        String sEncodingAESKey = WxUtil.RESP_MESSAGE_ENCODINGAESKEY;
        try {
            WXBizMsgCrypt wxcpt = new WXBizMsgCrypt(sToken, sEncodingAESKey,
sCorpID);

            String sEchoStr; //需要返回的明文
            sEchoStr = wxcpt.VerifyURL(signature, timestamp, nonce, echostr);
            System.out.println("verifyurl echostr: " + sEchoStr);
            //验证 URL 成功，将 sEchoStr 返回
            PrintWriter out = response.getWriter();
            out.write(sEchoStr);
            out.flush();

```

```

        out.close();
    } catch (Exception e) {
        //验证 URL 失败, 错误原因请查看异常
        e.printStackTrace();
    }
}

/**
 * 处理微信服务器发来的消息
 */
public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    //读取消息, 执行消息处理
    //微信加密签名
    String sReqMsgSig = request.getParameter("msg_signature");
    //时间戳
    String sReqTimeStamp = request.getParameter("timestamp");
    //随机数
    String sReqNonce = request.getParameter("nonce");
    String sToken = WxUtil.RESP_MESSAGE_TOKEN;
    String sCorpID = WxUtil.RESP_MESSAGE_CORPID;
    String sEncodingAESKey = WxUtil.RESP_MESSAGE_ENCODINGAESKEY;
    try {
        //POST 请求的密文数据
        //sReqData = HttpUtils.PostData();
        ServletInputStream in = request.getInputStream();
        BufferedReader reader = new BufferedReader(new InputStreamReader
(in));

        String sReqData="";
        String itemStr="";//作为输出字符串的临时串, 用于判断是否读取完毕
        while(null!=(itemStr=reader.readLine())){
            sReqData+=itemStr;
        }
        //对消息进行处理获得明文
        WXBizMsgCrypt wxcpt = new WXBizMsgCrypt(sToken, sEncodingAESKey, sCorpID);
        String sMsg = wxcpt.DecryptMsg(sReqMsgSig, sReqTimeStamp, sReqNonce, sReqData);
        //输出解密后的文件
        System.out.println("after decrypt msg: " + sMsg);
        //TODO: 解析出明文 XML 标签的内容进行处理
        //For example:
        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
        DocumentBuilder db = dbf.newDocumentBuilder();
        StringReader sr = new StringReader(sMsg);
        InputSource is = new InputSource(sr);
        Document document = db.parse(is);
        Element root = document.getDocumentElement();
    }
}

```

```

//PackageId,用于回复消息
NodeList nodelist_msgType = root.getElementsByTagName("PackageId");
String contentPackageId = nodelist_msgType.item(0).getTextContent();
System.out.println("-----contentPackageId:"+contentPackageId);
//信息节点个数
NodeList itemCount_note=root.getElementsByTagName("ItemCount");
long itemCount=Long.valueOf(itemCount_note.item(0).getTextContent()+"");
//Item 列表, 遍历信息节点
NodeList itemNodeRootList = root.getElementsByTagName("Item");
for (int i = 0; i < itemCount; i++) {
    //查看数据库 FromUserName+CreateTime 或 MsgID, 进行排重
    //重复消息, 直接结束本次循环, 进行下次循环 (continue)
    //数据库保存 FromUserName+CreateTime 或 MsgID
    NodeList itemNodeRoot = itemNodeRootList.item(i).getChildNodes();
    String msgType="";//消息类型
    for (int j = 0; j < itemNodeRoot.getLength(); j++) {
        if("MsgType".equals(itemNodeRoot.item(j).getNodeName()+"")){
            msgType=itemNodeRoot.item(j).getTextContent();
            break;
        }
    }
    if("event".equals(msgType)){
        //如果是事件
        executeEventMsg(itemNodeRoot);
    }else if("text".equals(msgType)){
        //如果是文本消息
    }else if("image".equals(msgType)){
        //如果是图片消息
    }else if("voice".equals(msgType)){
        //如果是声音消息
    }
}
//输出, 明文回复 PackageId
PrintWriter out = response.getWriter();
out.write(contentPackageId);
out.flush();
out.close();
} catch (Exception e) {
    e.printStackTrace();
}
}

/**
 * 处理事件消息
 * @param itemNodeRoot
 */

```



```

public static void executeEventMsg( NodeList itemNodeRoot){
    HashMap<String, String> map = new HashMap<String, String>();
    for (int j = 0; j < itemNodeRoot.getLength(); j++) {
        //放入 map 中
        map.put(itemNodeRoot.item(j).getNodeName(), itemNodeRoot.item(j).
getTextContent()+"");
    }
    //获得事件类型
    String eventType=map.get("Event");
    if(eventType.equals("create_chat")){
        //创建会话事件
        System.out.println("这是创建会话事件");
        String chatId="";//讨论组 ID
        String name="";//讨论组标题
        String owner="";//管理员 UserID
        String[] userArray=null;
        String userStr = "";
        //获得创建信息
        NodeList chatInfo;
        for (int j = 0; j < itemNodeRoot.getLength(); j++) {
            //放入 map 中
            if("ChatInfo".equals(itemNodeRoot.item(j).getNodeName())){
                chatInfo = itemNodeRoot.item(j).getChildNodes();
                //获得讨论组信息
                for (int i = 0; i < chatInfo.getLength(); i++) {
                    if("ChatId".equals(chatInfo.item(i).getNodeName()+
"")){
                        chatId=itemNodeRoot.item(i).getTextContent();
                    }else if("Name".equals(chatInfo.item(i).getNodeName()+
"")){
                        name=itemNodeRoot.item(i).getTextContent();
                    }else if("Owner".equals(chatInfo.item(i).
getNodeName()+"")){
                        owner=itemNodeRoot.item(i).getTextContent();
                    }else if("UserList".equals(chatInfo.item(i).
getNodeName()+"")){
                        //会话成员列表, 成员用 UserID 标识, 成员间以竖线 "|" 分隔
                        userStr=itemNodeRoot.item(i).getTextContent();
                        if(null!=userStr&&"!".equals(userStr)){
                            userArray=userStr.split("|");
                        }
                    }
                }
            }
        }
        break;
    }
}

```



```
        //信息处理,根据不同的企业 IM 接口进行处理
        System.out.println("chatId:"+chatId+" name:"+name+" owner:"+
owner+" userStr:"+userStr);
    }else if(eventType.equals("update_chat")){
        //修改会话人员
        System.out.println("这是修改会话人员事件");
    }else if(eventType.equals("quit_chat")){
        //退出会话
        System.out.println("这是退出会话事件");
    }else if(eventType.equals("subscribe")){
        //订阅事件,开启企业会话
        System.out.println("这是订阅事件");
    }else if(eventType.equals("unsubscribe")){
        //取消订阅事件,关闭企业会话
        System.out.println("这是取消订阅事件");
    }
}
```



备注:当接收到的微信客户端消息过多或者因其他原因无法及时回应时,需要回复空消息,进行异步处理。

通讯录管理及异步任务

通过对前面几章的学习，可以发现每个 API 接口都需要成员，那成员从何而来？又在哪里维护？部门树又该如何导入？本章将带领读者学习企业号开发中通讯录成员的导入与维护、标签的管理，以及如何通过异步任务维护通讯录部门、成员等。

本章主要涉及的知识点有：

- 验证关注：成员可以通过何种方式关注，关注条件及如何进行二次验证。
- 部门管理：学习如何通过接口实现企业号中部门的查看、创建、修改以及删除。
- 成员管理：学习企业号中成员的查看、创建、修改以及删除。
- 标签管理：学习如何通过接口实现标签的管理，利用标签实现不同人员的应用权限。
- 异步任务：了解异步任务的基础知识，学习如何通过异步任务维护企业号通讯录。
- 接口的实际应用：通过本章最后的案例，学习通讯录管理的各接口运用，学习如何维护企业号通讯录。

7.1 成员验证关注

成员要想使用企业号，首先需要成功关注企业号，在成员关注企业号时，将根据企业号默认应用“通讯录”中维护的成员微信号、手机、邮箱（三个中任意一个即可），与成员的微信号、微信绑定的手机或邮箱进行匹配，匹配成功，成员即可关注。目前，成员关注方式主要有三种：

- 通过二维码扫描关注。
- 通过“推荐给朋友”的方式进行关注，如图 7.1 所示。
- 通过关键词“企业号”快速搜索，开启“企业号搜索”之后，成员可能通过微信搜索栏，输入“企业号”+企业号全称的方式进行搜索，



备注：对于企业号主动邀请的方式，由于较多用户收到不良营销等企业号的骚扰，因此微信取消了主动邀请功能。

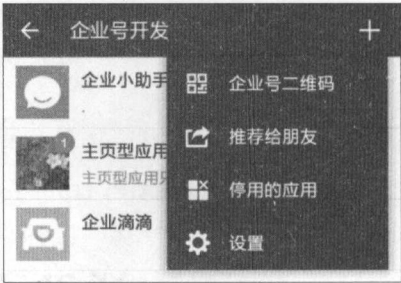


图 7.1 通过“推荐给朋友”关注企业号

7.2 部门管理

通讯录的维护建议与企业部门结构相同，可以通过异步任务进行批量统一维护。本节将简单介绍单个部门的维护，在部门创建过程中，需要注意部门总数不能超过 3 万，最大层级 15 层。

7.2.1 新增部门

通过接口创建部门说明如下。

(1) Https 请求链接：

https://qyapi.weixin.qq.com/cgi-bin/department/create?access_token=ACCESS_TOKEN

(2) 数据请求方式。

POST 方式进行数据请求。

(3) 消息请求结构体：

```
{
  "name": "分公司",
  "parentid": 1,
  "order": 1,
  "id": 2
}
```

(4) 参数说明。详细说明如表 7.1 所示。

表 7.1 创建部门请求消息体参数说明

参数	是否必需	说明
access_token	是	调用接口凭证
name	是	部门名称。长度限制为1~64字节，字符不能包括\:*?"<>
parentid	是	父部门ID。根部门ID为1
order	否	在父部门中的次序值，order值小的排序靠前
id	否	部门ID，整型。指定时必须大于1，不指定时则自动生成

注意：微信的部门根目录是从 1 开始，所以需要针对自己的部门进行相应的修改。
access_token 为主动模式下的接口使用票据，需要进行缓存处理（在 3.2、3.3 节介绍），本章将不再介绍。

(5) 返回参数及说明：

```
{
  "errcode": 0,
  "errmsg": "created",
  "id": 2
}
```

返回参数中，errcode 值为 0 则表示执行成功，其他结果均为失败。ID 为成功创建部门之后，返回的部门 ID，主要用于部门自动生成的场景。

7.2.2 更新部门

通过接口更新部门说明如下。

(1) Https 请求链接：

https://qyapi.weixin.qq.com/cgi-bin/department/update?access_token=ACCESS_TOKEN

(2) 数据请求方式。

Post 方式进行数据请求。

(3) 消息请求结构体：

```
{
  "id": 2,
  "name": "改名研发中心",
  "parentid": 1,
  "order": 1
}
```

(4) 参数说明。详细说明如表 7.2 所示。

表 7.2 更新部门请求消息体参数说明

参数	是否必需	说明
access_token	是	调用接口凭证
name	是	部门名称。长度限制为1~64字节，字符不能包括\:*?"<>
parentid	是	父部门ID。根部门ID为1
order	否	在父部门中的次序值，order值小的排序靠前
id	否	部门ID，整型。指定时必须大于1，不指定时则自动生成

注意：更新部门时需要注意部门 ID，勿错误更新部门。

(5) 返回参数及说明:

```
{
  "errcode": 0,
  "errmsg": "updated"
}
```

errcode 为执行结果，0 表示成功，1 表示失败。

7.2.3 删除部门

通过接口删除部门说明如下。

(1) Https 请求链接:

https://qyapi.weixin.qq.com/cgi-bin/department/delete?access_token=ACCESS_TOKEN&id=ID


(2) 数据请求方式。

GET 方式进行数据请求。

(3) 参数说明。详细说明如表 7.3 所示。

表 7.3 删除部门请求消息体参数说明

参数	是否必需	说明
access_token	是	调用接口凭证
id	否	部门ID

 备注：不能删除根部门，不能删除含有子部门或者成员的部门。

(4) 返回参数及说明:

```
{
  "errcode": 0,
  "errmsg": "deleted"
}
```

errcode 为执行结果，0 表示成功，1 表示失败。

7.2.4 获取部门列表

根据输入的部门 ID，通过接口获得当前部门及子部门的信息，接口说明如下。

(1) Https 请求链接:

https://qyapi.weixin.qq.com/cgi-bin/department/list?access_token=ACCESS_TOKEN&id=ID

(2) 数据请求方式。

GET 方式进行数据请求。

(3) 参数说明。详细说明如表 7.4 所示。

表 7.4 获取部门列表请求消息体参数说明

参数	是否必需	说明
access_token	是	调用接口凭证
id	否	部门ID

(4) 返回参数及说明:

```
{
  "errcode": 0,
  "errmsg": "ok",
  "department": [
    {
      "id": 2,
      "name": "改名研发中心",
      "parentid": 1,
      "order": 10
    },
    {
      "id": 3,
      "name": "产品部",
      "parentid": 2,
      "order": 40
    }
  ]
}
```

errcode 为执行结果，0 表示成功，1 表示失败，参数详细说明如表 7.5 所示。

表 7.5 获取部门列表响应消息参数说明

参数	说明
errcode	返回码
errmsg	对返回码的文本描述内容
department	部门列表数据。按部门的order字段从小到大排列
id	部门ID
name	部门名称
parentid	父部门ID。根部门为1
order	在父部门中的次序值。order值小的排序靠前

7.3 成员管理

成员的维护与部门维护相同，建议通过异步任务进行统一的管理维护。本节将简单介绍成员维护的各个接口。

7.3.1 新增成员

通过接口能够实现新的成员添加，接口详细说明如下。

(1) Https 请求链接:

https://qyapi.weixin.qq.com/cgi-bin/user/create?access_token=ACCESS_TOKEN

(2) 数据请求方式:

POST 方式进行数据请求。


(3) 消息请求结构体:

```
{
  "userid": "muyunfei",
  "name": "牟云飞",
  "department": [1, 2],
  "position": "产品经理",
  "mobile": "1556257xxxx",
  "gender": "1",
  "email": "1147417467@qq.com",
  "weixinid": "GeneralMou",
  "avatar_mediaid": "2-G6nrLmr5EC3MNB_-zLldDdzkd0p7cNliYu9V5w7o8K0",
  "extattr": {"attrs":[{"name":"爱好","value":"旅游"},
    {"name":"卡号","value":"1234567234"}]}
}
```

(4) 参数说明。详细说明如表 7.6 所示。

表 7.6 新增成员请求消息体参数说明

参数	是否必需	说明
access_token	是	调用接口凭证
userid	是	成员UserID。对应管理端的账号，企业内必须唯一。不区分大小写，长度为1~64字节
name	是	成员名称。长度为1~64字节
department	是	成员所属部门ID列表，不超过20个
position	否	职位信息。长度为0~64字节
mobile	至少填一个	手机号码。企业内必须唯一，mobile/weixinid/email三者不能同时为空
email		邮箱。长度为0~64字节。企业内必须唯一
weixinid		微信号。企业内必须唯一
gender	否	性别。1表示男性，2表示女性
avatar_mediaid	否	成员头像的MediaId，通过多媒体接口上传图片获得的MediaId
extattr	否	扩展属性。将在7.3.2节中介绍

 **注意：**weixinid 是微信号，不是微信的名字。由于企业人员可能身兼多个职务，所以成员所属部门 department 是部门 ID 列表，并且使用逗号进行分隔。

(5) 返回参数及说明:

```
{
  "errcode": 0,
  "errmsg": "created",
}
```

返回参数中, errcode 值为 0 则表示执行成功, 其他结果均为失败, 详见返回码参见附录 B。

7.3.2 成员扩展属性 extattr

成员扩展属性 extattr, 在成员新增、修改、异步任务中都可以就行维护, 但在使用之前, 需要在 Web 管理端创建后才能生效, 否则忽略未知属性的赋值。本节将演示扩展属性的创建。

01 登录企业号 Web 管理端, 打开“通讯录”, 单击“设置”按钮, 如图 7.2 所示。

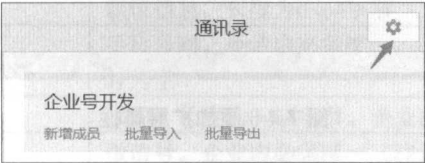


图 7.2 成员创建扩展属性

02 进入“通讯录”设置页面之后, 单击“扩展字段”, 如图 7.3 所示。

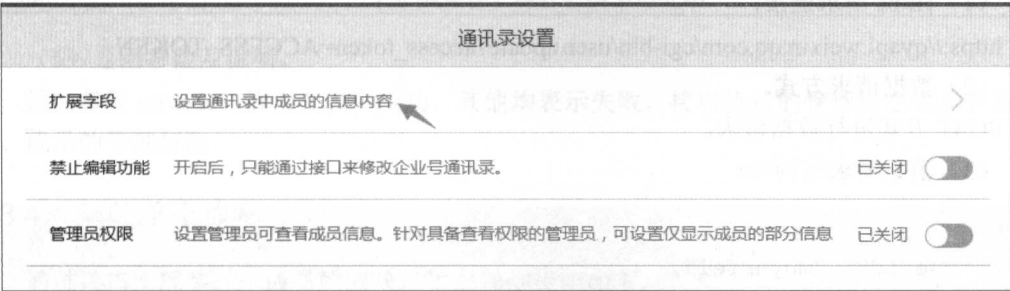


图 7.3 进入扩展字段维护界面

03 扩展字段分为普通字段和唯一性字段, 如图 7.4 所示。唯一性字段的内容在通讯录中不允许重复, 需要保证数据的唯一性, 而普通字段允许数据的重复, 方便读者实现不同业务场景。

备注: 在多个微信账号对应一个系统账号的场景下, 除了自身数据库进行转换之外, 还可以通过新增普通字段实现业务。

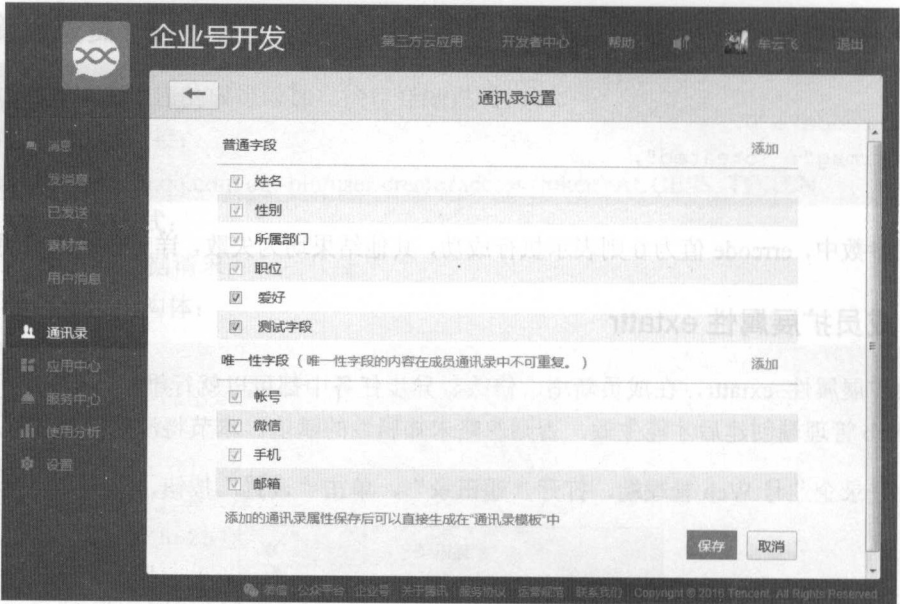


图 7.4 添加扩展属性

7.3.3 维护成员信息

通过接口能够实现新的成员添加，接口详细说明如下。

(1) Https 请求链接:

https://qyapi.weixin.qq.com/cgi-bin/user/update?access_token=ACCESS_TOKEN

(2) 数据请求方式。

POST 方式进行数据请求。

(3) 消息请求结构体:

```
{
  "userid": "muyunfei",
  "name": "牟云飞",
  "department": [1],
  "position": "产品经理",
  "mobile": "1556257xxxx",
  "gender": "1",
  "email": "1147417467@qq.com",
  "weixinid": "GeneralMou",
  "enable": 1,
  "avatar_mediaid": "2-G6nrLmr5EC3MNB_-zL1dDdzkd0p7cNliYu9V5w7o8K0",
  "extattr": {
    "attrs": [
      { "name": "爱好", "value": "旅游" },
      { "name": "卡号", "value": "1234567234" }
    ]
  }
}
```



备注：维护成员信息与新增成员所包含的请求信息基本一致，增加 enable 禁用属性，对于不需要进行变更的属性信息，可以不发送该属性。扩展属性 extattr、name 为 Web 端创建的字段名，value 为字段值。

(4) 参数说明。详细说明如表 7.7 所示。

表 7.7 新增成员请求消息体参数说明

参数	是否必需	说明
access_token	是	调用接口凭证
userid	是	成员UserID。对应管理端的账号，企业内必须唯一。不区分大小写，长度为1~64字节
name	否	成员名称。长度为1~64字节
department	否	成员所属部门ID列表，不超过20个
position	否	职位信息。长度为0~64字节
mobile	否	手机号码。企业内必须唯一，mobile/weixinid/email三者不能同时为空
email	否	邮箱。长度为0~64字节。企业内必须唯一
weixinid	否	微信号。企业内必须唯一（注意：是微信号，不是微信的名字）
enable	否	启用/禁用成员。1表示启用成员，0表示禁用成员
gender	否	性别。1表示男性，2表示女性
avatar_mediaid	否	成员头像的MediaId，通过多媒体接口上传图片获得的MediaId
extattr	否	扩展属性。将在7.3.2节中介绍

(5) 返回参数及说明。

返回参数 errcode，值为 0 表示成功，其他均表示失败，接口执行的管理组必须拥有指定部门、成员的管理权限。

7.3.4 删除单个成员

通过接口实现某一个成员的删除，接口详细说明如下。

(1) Https 请求链接：

https://qyapi.weixin.qq.com/cgi-bin/user/delete?access_token=ACCESS_TOKEN&userid=

(2) 数据请求方式。

GET 方式进行数据请求。

(3) 请求参数说明如表 7.8 所示。

表 7.8 删除单个成员请求消息体参数说明

参数	是否必需	说明
access_token	是	调用接口凭证
userid	是	成员UserID。对应管理端的账号，企业内必须唯一

(4) 返回参数及说明：

返回参数 errcode，值为 0 表示成功，其他均表示失败。

7.3.5 批量删除成员

通过接口能够实现新的成员添加，接口详细说明如下。

(1) Https 请求链接。


https://qyapi.weixin.qq.com/cgi-bin/user/batchdelete?access_token=ACCESS_TOKEN

(2) 数据请求方式：

POST 方式进行数据请求。

(3) 消息请求结构体：

```
{
  "useridlist": ["zhangsan", "lisi"]
}
```

 备注：多个人员之间用逗号分隔。

(4) 请求参数说明，如表 7.9 所示。

表 7.9 批量删除成员请求消息体参数说明

参数	是否必需	说明
access_token	是	调用接口凭证
userid	是	成员UserID。对应管理端的账号，企业内必须唯一

(5) 返回参数及说明。

返回参数 errcode，值为 0 表示成功，其他均表示失败。

7.3.6 获取成员信息

通过接口获得成员信息，读者能够与订阅事件相结合，实现关注成员的头像等信息的回传同步，接口详细说明如下。

(1) Https 请求链接：

https://qyapi.weixin.qq.com/cgi-bin/user/get?access_token=ACCESS_TOKEN&userid=USERID

(2) 数据请求方式：

GET 方式进行数据请求。

(3) 请求参数说明，如表 7.10 所示。

表 7.10 获取成员信息请求消息体参数说明

参数	是否必需	说明
access_token	是	调用接口凭证
userid	是	成员UserID，对应管理端的账号

(4) 返回数据参数：

```
{
  "errcode": 0,
```

```

    "errmsg": "ok",
    "userid": "muyunfei",
    "name": "牟云飞",
    "department": [1, 2],
    "position": "后台工程师",
    "mobile": "1556257xxxx",
    "gender": "1",
    "email": "zhangsan@gzdev.com",
    "weixinid": "lisifordev",
    "avatar":
"http://wx.qlogo.cn/mmopen/ajNVdqHZLLA3WJ6DSZUfiakYe37PKnQhBIeQQB04czqrnZDS7
9FH5Wm5m4X69TBicnHFlhiafvDwklOpZeXYQQ2icg/0",
    "status": 1,
    "extattr": {"attrs":[{"name":"爱好","value":"旅游"}, {"name":"卡号
", "value":"1234567234"}]}
}

```

(5) 返回参数说明如表 7.11 所示。

表 7.11 获取成员信息返回消息体参数说明

参数	说明
errcode	返回码
errmsg	对返回码的文本描述内容
userid	成员UserID。对应管理端的账号
name	成员名称
department	成员所属部门ID列表
position	职位信息
mobile	手机号码
gender	性别。0表示未定义，1表示男性，2表示女性
email	邮箱
weixinid	微信号
avatar	头像URL。注：如果要获取小图，则将URL最后的"/0"改成"/64"即可
status	关注状态： 1=已关注，2=已禁用，4=未关注
extattr	扩展属性

(6) 示例代码：

```

/**
 * 获取通讯录成员信息
 * @param userid  查询的账户 ID
 * @return
 * @throws IOException
 */
public JSONObject getUserMsg(String userid)
{
    //获取微信号接口调用票据

```



```
String token=getTokenFromWx();
try {
    CloseableHttpClient httpClient = HttpClients.createDefault();
    HttpPost httpPost= new HttpPost("https://qyapi.weixin.qq.com/cgi-bin/user/get?access_token="+token+"&userid="+userid);
    //自定义请求响应
    ResponseHandler<JSONObject> responseHandler = new ResponseHandler<JSONObject>() {
        public JSONObject handleResponse(
            final HttpResponse response) throws ClientProtocolException,
            IOException {
            int status = response.getStatusLine().getStatusCode();
            if (status >= 200 && status < 300) {
                HttpEntity entity = response.getEntity();
                if(null!=entity){
                    String result= EntityUtils.toString(entity);
                    //根据字符串生成 JSON 对象
                    JSONObject resultObj = JSONObject.fromObject(result);
                    return resultObj;
                }else{
                    return null;
                }
            } else {
                throw new ClientProtocolException("Unexpected response status:
" + status);
            }
        }
    };
    //返回的 JSON 对象
    JSONObject responseBody = httpClient.execute(httpPost, responseHandler);
    return responseBody;
}catch (Exception e) {
    e.printStackTrace();
    return null;
}
}
```



备注: getTokenFromWx()为获取接口调用票据的方法, 可以查看 3.3 节。

7.3.7 获取部门成员

获取当前部门或当前部门及子部门人员, 接口详细说明如下。

(1) Https 请求链接:

https://qyapi.weixin.qq.com/cgi-bin/user/simplelist?access_token=ACCESS_TOKEN&department_id=DEPARTMENT_ID&fetch_child=FETCH_CHILD&status=STATUS

- (2) 数据请求方式。
GET 方式进行数据请求。
- (3) 请求参数说明如表 7.12 所示。

表 7.12 获取部门成员请求消息体参数说明

参数	是否必需	说明
access_token	是	调用接口凭证
department_id	是	获取的部门ID
fetch_child	否	1/0：是否递归获取子部门下面的成员
status	否	0获取全部成员，1获取已关注成员列表，2获取禁用成员列表，4获取未关注成员列表。status可叠加，未填写则默认为4

- (4) 返回数据参数：

```
{
  "errcode": 0,
  "errmsg": "ok",
  "userlist": [
    {
      "userid": "zhangsan",
      "name": "李四",
      "department": [1, 2]
    }
  ]
}
```

参数详细说明如表 7.13 所示。

表 7.13 获取部门成员信息返回消息体参数说明

参数	说明
errcode	返回码
errmsg	对返回码的文本描述内容
userlist	成员列表
userid	成员UserID。对应管理端的账号
name	成员名称
department	成员所属部门

7.3.8 获取部门成员及详细信息

获取当前部门成员及成员详细资料信息，接口详细说明如下。

- (1) Https 请求链接：

```
https://qyapi.weixin.qq.com/cgi-bin/user/list?access_token=ACCESS_TOKEN&department_id=DEPARTMENT_ID&fetch_child=FETCH_CHILD&status=STATUS
```


- (2) 数据请求方式：
GET 方式进行数据请求。
- (3) 请求参数说明如表 7.14 所示。

表 7.14 获取部门成员及详细信息请求消息体参数说明

参数	是否必需	说明
access_token	是	调用接口凭证
department_id	是	获取的部门ID
fetch_child	否	1/0: 是否递归获取子部门下面的成员
status	否	0获取全部成员, 1获取已关注成员列表, 2获取禁用成员列表, 4获取未关注成员列表。status可叠加, 未填写则默认为4

- (4) 返回数据参数:

```
{
  "errcode": 0,
  "errmsg": "ok",
  "userlist": [
    {
      "userid": "muyunfei",
      "name": "牟云飞",
      "department": [1, 2],
      "position": "后台工程师",
      "mobile": "1556257xxxx",
      "gender": "1",
      "email": "zhangsan@qq.com",
      "weixinid": "lisifordev",
      "avatar":
"http://wx.qlogo.cn/mmopen/ajNVdqHZLLA3WJ6DSZUfiakYe37PKnQhBIeOQB04czqrnZDS7
9FH5Wm5m4X69TBicnHFlhiafvDwklOpZeXYQQ2icg/0",
      "status": 1,
      "extattr": { "attrs": [{"name": " 爱好 ", "value": " 旅 游
"}, {"name": "卡号", "value": "1234567234"}] }
    }
  ]
}
```

 **使用技巧：**数据查询数据量较大，可以通过订阅事件或定时任务同步信息到读者数据库，后期信息查询查询自身数据库即可。

参数详细说明如表 7.15 所示。

表 7.15 获取部门成员及详细信息返回消息体参数说明

参数	说明
errcode	返回码

续表

参数	说明
errmsg	对返回码的文本描述内容
userlist	成员列表
userid	成员UserID。对应管理端的账号
name	成员名称
department	成员所属部门ID列表
position	职位信息
mobile	手机号码
gender	性别。0表示未定义，1表示男性，2表示女性
email	邮箱
weixinid	微信号
avatar	头像URL。注：如果要获取小图，则将URL最后的"/0"改成"/64"即可
status	关注状态：1=已关注，2=已冻结，4=未关注
extattr	扩展属性

7.4 异步任务管理

异步任务是企业号通讯录维护中较为重要的方式，是在批量更新部门或者人员时，通过“全量覆盖部门接口”和“全量覆盖成员接口”，向微信服务器发送一个 CSV 的文件。微信服务器会根据 CSV 文件进行更新，将企业部门、职员等信息批量推送至企业号。企业号在后台进行异步处理，并将最终执行结果返回读者管理端，非常方便地构建与企业相同的组织部门架构，实现企业部门、职员的信息同步。

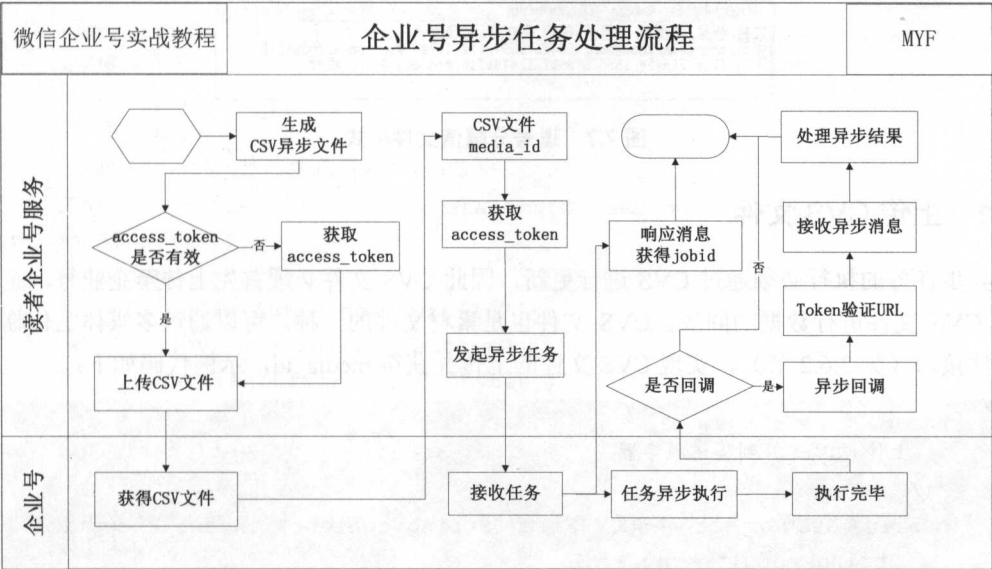


图 7.5 异步任务处理流程

异步任务处理流程包含主动调用及消息回调，处理方式较为复杂（如图 7.5 所示），主要

分为以下三步。

01 读者下载模板文件，文件后尾名为.csv（逗号分隔值文件，以纯文本形式存储表格数据，读者可以使用 Excel 打开，如图 7.6 所示），如图 7.7 所示，按照模板的格式生成接口所需的 CSV 格式的文件流，利用有效的 access_token 上传至微信服务器得到 media_id，注意换行使用\n。

▶ 备注：每个接口的 CSV 文件不同，读者需要根据接口进行下载，其次对于读者自定义的成员扩展属性 extattr（7.3.2 节中介绍），需要读者在模板文件中自行添加。注意，扩展属性名字与企业号 Web 界面管理端名字保持一致。

02 利用 media_id 向企业号发送异步任务消息，通知企业号执行异步任务，并获得异步执行任务 ID（jobid，用于查询异步任务状况，也可以不查询，等待企业号异步任务执行成功后通知）。

03 接收异步任务完成的回调消息，获取任务执行结果。

注意：读者发送异步任务时，如果不填写回调链接，将不进行异步通知。

	A	B	C	D	E	F	G
1	姓名	帐号	微信号	手机号	邮箱	所在部门	职位
2	张三	zhangsan	zhangsan	15562579597	1147417467@qq.com	4	产品经理
3	李四	lisi	lisi14	15562579597	1147417467@qq.com	6,7	工程师
4							

图 7.6 Excel 打开 CVS 文件

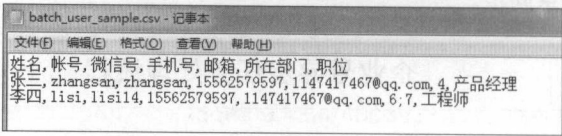


图 7.7 逗号分隔值文件格式

7.4.1 上传 CVS 文件

异步任务的执行必须通过 CVS 进行更新，因此 CVS 文件必须首先上传至企业号，企业号根据 CVS 文件进行数据的同步。CVS 文件也是素材文件的一种，可以通过多媒体上传临时素材文件接口（见 3.6.2 节），实现 CVS 文件的上传并获得 media_id，示例代码如下：

```
/**
 * 上传 CSV 文件到腾讯服务器
 */
public JSONObject sendCVSFile( String content) throws Exception {
    String result = null;
    String token=getTokenFromWx();
    System.out.println("token:"+token);
}
/**
```

```

* 第一部分
*/
URL urlObj = new URL("https://qyapi.weixin.qq.com/cgi-bin/media/upload?access_token="+ token + "&type=file");
URLConnection con = (URLConnection) urlObj.openConnection();
con.setRequestMethod("POST"); //以 POST 方式提交表单, 默认是 GET 方式
con.setDoInput(true);
con.setDoOutput(true);
con.setUseCaches(false); //POST 方式不能使用缓存
//设置请求头信息
con.setRequestProperty("Connection", "Keep-Alive");
con.setRequestProperty("Charset", "UTF-8");
//设置边界
String BOUNDARY = "-----" + System.currentTimeMillis();
con.setRequestProperty("Content-Type", "multipart/form-data;boundary="+ BOUNDARY);
//请求正文信息
//第一部分:
StringBuilder sb = new StringBuilder();
sb.append("--"); //必须多两道线
sb.append(BOUNDARY);
sb.append("\r\n");
sb.append("Content-Disposition:form-data;name=\"media\";filename=\""+ "info.csv" + "\"\r\n");
sb.append("Content-Type:application/octet-stream\r\n\r\n");
byte[] head = sb.toString().getBytes("utf-8");
//获得输出流
OutputStream out = new DataOutputStream(con.getOutputStream());
//输出表头
out.write(head);
//文件正文部分
//把文件以流文件的方式 推入到 URL 中
DataInputStream in = new DataInputStream(new ByteArrayInputStream(content.getBytes()));
int bytes = 0;
byte[] bufferOut = new byte[1024];
while ((bytes = in.read(bufferOut)) != -1) {
    out.write(bufferOut, 0, bytes);
}
in.close();
// 结尾部分
byte[] foot = ("\r\n--" + BOUNDARY + "--\r\n").getBytes("utf-8");
// 定义最后数据分隔线
out.write(foot);
out.flush();
out.close();

```

```
StringBuffer buffer = new StringBuffer();
BufferedReader reader = null;
try {
    // 定义 BufferedReader 输入流来读取 URL 的响应
    reader = new BufferedReader(new InputStreamReader
(con.getInputStream()));
    String line = null;
    while ((line = reader.readLine()) != null) {
        buffer.append(line);
    }
    if(result==null){
        result = buffer.toString();
    }
} catch (IOException e) {
    System.out.println("发送 POST 请求出现异常! " + e);
    e.printStackTrace();
    throw new IOException("数据读取异常");
} finally {
    if(reader!=null){
        reader.close();
    }
}
JSONObject jsonObj =JSONObject.fromObject(result);
return jsonObj;
}
```



备注：与多媒体文件上传方法一致，均是通过流的形式进行上传，唯一不同是指定头部信息 filename 为 CVS 文件。

7.4.2 全量覆盖部门

全量覆盖部门能够将提交的 CVS 文件与企业号通讯录组织架构保持一致，实现企业组织架构的统一，其执行内容包括：

- 文件中存在、通讯录中也存在的部门，执行修改操作。
- 文件中存在、通讯录中不存在的部门，执行添加操作。
- 文件中不存在、通讯录中存在的部门，当部门下没有任何成员或子部门时执行删除操作。

CSV 文件中，部门名称、部门 ID、父部门 ID 为必填字段，部门 ID 必须为数字，排序为可选字段，置空或填 0 时不修改排序。



注意：部门根节点为 1，需要对根节点进行特殊处理。列数据使用逗号分隔，换行使用\n。

接口操作详细说明如下。

- (1) Https 请求链接：

https://qyapi.weixin.qq.com/cgi-bin/batch/replaceparty?access_token=ACCESS_TOKEN

(2) 数据请求方式。

POST 方式进行数据请求。

(3) CVS 文件格式：

部门名称, 部门 ID, 父部门 ID, 排序				
总办	2	1	1	
事业群	3	1	2	
产品部	4	3	3	
开发部	5	3	4	

如果需要下载已上传的 CVS 文件并进行查看时，可以使用 Excel 打开，如图 7.8 所示，能够更清晰地查看 CVS 部门数据。

	A	B	C	D	E
1	部门名称	部门ID	父部门ID	排序	
2	总办	2	1	1	
3	事业群	3	1	2	
4	产品部	4	3	3	
5	开发部	5	3	4	
6					
7					

图 7.8 全量覆盖部门 CVS 文件

(4) 请求消息结构体及参数说明：

```
{
  "media_id": "xxxxxxx",
  "callback": {
    "url": "xxx",
    "token": "xxx",
    "encodingaeskey": "xxx"
  }
}
```

参数详细说明如表 7.16 所示。

表 7.16 全量覆盖部门请求消息体参数说明

参数	是否必需	说明
media_id	是	上传的CSV文件的media_id
callback	否	回调信息。如填写该项，则任务完成后，通过callback推送事件给企业
url	否	企业应用接收企业号推送请求的访问协议和地址，支持HTTP或HTTPS协议
token	否	用于生成签名
encodingaeskey	否	用于消息体的加密，是AES密钥的Base64编码



备注：callback 中的参数为读者企业号服务的链接地址，是异步任务执行完成之后，通知读者的通道。其链接处理方式与被动回调模式开启方式一致，需要验证 URL 的有效性，token、encodingaeskey 为开启回调模式中的参数值。

(5) 返回结果及参数说明：

```
{
    "errcode": 0,
    "errmsg": "ok",
    "jobid": "xxxxxx"
}
```

返回参数详细说明如表 7.17 所示。

表 7.17 全量覆盖部门请求返回参数说明

参数	说明
errcode	返回码
errmsg	对返回码的文本描述内容
jobid	异步任务ID，最大长度为64字节



备注：异步任务除了在任务执行完毕时推送通知外，还可以主动查询任务执行情况，其中，jobid 为异步任务的执行 ID，通过 jobid 能够实现异步任务执行情况的查询，详细说明将在 7.4.4 节中介绍。

(6) 示例代码：

```
//CVS 文件内容
StringBuffer content = new StringBuffer("部门名称,部门 ID,父部门 ID,排序")
    .append("\n");
content.append("总办,2,1,1").append("\n");
content.append("事业群,3,1,2").append("\n");
content.append("产品部,4,3,3").append("\n");
content.append("研发部,5,3,4").append("\n");
//上传 CSV 文件到腾讯服务器，获得 media_id
JSONObject json = util.sendCVSFile(content.toString());
System.out.println(json.toString());
String mediaId=json.getString("media_id");
//数据同步，发送异步任务请求
if(null!=mediaId&&!"".equals(mediaId)){
    //获得 token
    String token=util.getTokenFromWx();
    String url="https://qyapi.weixin.qq.com/cgi-bin/batc"+
        "h/replaceparty?access_token="+token;
    sendCVSData(mediaId ,url);
}
```



注意：这里不是在服务器生成 CVS 文件，而是直接使用 CVS 格式的字符串以数据流的形式进行提交。

封装 CVS 文件发送方法 `sendCVSData(String mediaId,String url)`，通过传递 `media_id` 以及 URL 实现异步任务的申请，示例代码如下：


```
public boolean sendCVSData(String mediaId,String url){
    String jsonContext="{\"+
        \"\\\"media_id\\\":\\\""+mediaId+\"\\\", \"+
        \"\\\"callback\\\": \"+
        \"{\"+
            \"\\\"url\\\": \\\""+WxUtil.webUrl+\"/getJbidServlet\"+\"\\\", \"+
            \"\\\"token\\\": \\\""+WxUtil.RESP_MESSAGE_TOKEN+\"\\\", \"+
            \"\\\"encodingaeskey\\\": \\\""+WxUtil.RESP_MESSAGE_
ENCODINGAESKEY+\"\\\""+
        \"}\"+
    \"}\";
    //发送消息
    //消息为 JSON 格式
    boolean flag=false;
    try {
        CloseableHttpClient httpclient = HttpClients.createDefault();
        HttpPost httpPost= new HttpPost(url);
        //发送 JSON 格式的数据
        StringEntity myEntity = new StringEntity(jsonContext,
            ContentType.create("text/plain", "UTF-8"));
        httpPost.setEntity(myEntity);
        //创建响应处理类
        ResponseHandler<JSONObject> responseHandler = new ResponseHandler
<JSONObject>() {
            public JSONObject handleResponse(
                final HttpResponse response) throws ClientProtocolException,
IOException {
                int status = response.getStatusLine().getStatusCode();
                if (status >= 200 && status < 300) {
                    HttpEntity entity = response.getEntity();
                    if(null!=entity){
                        String result= EntityUtils.toString(entity);
                        //根据字符串生成 JSON 对象
                        JSONObject resultObj = JSONObject.fromObject
(result);
                        return resultObj;
                    }else{
                        return null;
                    }
                } else {
```

```

        throw new ClientProtocolException("Unexpected response status:
" + status);
    }
}

};
//返回的 JSON 对象
JSONObject responseBody = httpclient.execute(httpPost,
responseHandler);
System.out.println(responseBody);
int result= (Integer) responseBody.get("errcode");
if(0==result){
    flag=true;
}else{
    flag=false;
}
httpclient.close();
} catch (Exception e) {
// TODO Auto-generated catch block
e.printStackTrace();
return false;
}
return true;


```

 备注：回调链接 getJbidServlet 将在 7.4.5 节中介绍。

7.4.3 全量覆盖成员

全量覆盖成员能够将提交的 CVS 文件与企业号通讯录内成员保持一致，实现企业人员的信息同步。CVS 文件同步信息包括：姓名、账号、微信号、手机号、邮箱、所在部门、职位、爱好（自定义信息）。账号（UserID）为企业号中成员的主键，微信号、手机号、邮箱三项必须填写一项，且三项中任意一项不能出现重复数据（不能出现相同的手机号、邮箱及微信号），其他注意内容包括：

- 模板中的部门需要填写部门 ID，多个部门之间使用分号进行分隔，且部门 ID 必须为数字。
- 文件中存在且通讯录中也存在的成员，完全以文件为准。
- 文件中存在且通讯录中不存在的成员，将执行添加操作。
- 通讯录中存在且文件中不存在的成员，将执行删除操作。
- 如果删除的成员多于 50 人，且多于现有人数的 20% 以上的异步任务将不予以导入。
- 如果删除的成员少于 50 人，且多于现有人数的 80% 以上的异步任务将不予以导入。
- 文件中如果有指定的字段，则以指定的字段值为准，文件中未指定的字段则不执行更新。

 备注：自定义信息是读者在 7.3.2 节中自定义的扩展属性 extattr，没有自定义属性的读者不需要添加该字段。

全量覆盖成员接口操作详细说明如下。

(1) Https 请求链接：

https://qyapi.weixin.qq.com/cgi-bin/batch/replaceuser?access_token= ACCESS_TOKEN

(2) 数据请求方式。

POST 方式进行数据请求。

(3) CVS 文件格式：

姓名,账号,微信号,手机号,邮箱,所在部门,职位
张三,zhangsan,zhangsan,1556257xxxx,1147417467@qq.com,4,产品经理
李四,lisi,lisi14,15562579999,123456789@qq.com,6;7,工程师

同全量覆盖部门一样，如果需要下载已上传的 CVS 文件并进行查看，则可以使用 Excel 打开，如图 7.9 所示，能够更清晰地查看 CVS 成员信息。

	A	B	C	D	E	F	G
1	姓名	帐号	微信号	手机号	邮箱	所在部门	职位
2	张三	zhangsan	zhangsan	15562579597	1147417467@qq.com	4	产品经理
3	李四	lisi	lisi14	15562579999	123456789@qq.com	6,7	工程师
4							
5							
6							

图 7.9 全量覆盖成员 CVS 文件

(4) 请求消息结构体及参数说明：

```
{
  "media_id": "xxxxxxx",
  "callback":
  {
    "url": "xxx",
    "token": "xxx",
    "encodingaeskey": "xxx"
  }
}
```

参数详细说明如表 7.18 所示。

表 7.18 全量覆盖成员请求消息体参数说明

参数	是否必需	说明
media_id	是	上传的CSV文件的media_id
callback	否	回调信息。如填写该项，则任务完成后，通过callback推送事件给企业
url	否	企业应用接收企业号推送请求的访问协议和地址，支持HTTP或HTTPS协议
token	否	用于生成签名
encodingaeskey	否	用于消息体的加密，是AES密钥的Base64编码



备注：callback 中的参数为读者企业号服务的链接地址，是异步任务执行完成之后，通知读者的通道。其链接处理方式与被动回调模式开启方式一致，需要验证 URL 的有效性，token、encodingaeskey 为开启回调模式中的参数值。

(5) 返回结果及参数说明：

```
{
    "errcode": 0,
    "errmsg": "ok",
    "jobid": "xxxxx"
}
```

返回参数详细说明如表 7.19 所示。

表 7.19 全量覆盖部门请求返回参数说明

参数	说明
errcode	返回码
errmsg	对返回码的文本描述内容
jobid	异步任务ID，最大长度为64字节




备注：异步任务除了在任务执行完毕时推送通知外，还可以主动查询任务执行情况，其中 jobid 为异步任务的执行 ID，通过 jobid 能够实现异步任务执行情况的查询，详细说明将在 7.4.4 节中介绍。

(6) 示例代码：


```
//CVS 文件内容
StringBuffer cvs_contacts=new StringBuffer("姓名,账号,微信号,手机号,邮箱,
性别,所在部门,职位").append("\n");
cvs_contacts.append("张三,zhangsan,zhangsan,1556257xxxx,1147417467@qq.com,
1,4,产品经理").append("\n");
cvs_contacts.append("李四,lisi,lisi4,15562579999,1556257xxxx@163.com,
1,1,高级研发工程师").append("\n");
//上传 CSV 文件到腾讯服务器,获得 media_id
JSONObject json = util.sendCVSFile(cvs_contacts.toString());
System.out.println(json.toString());
String mediaId=json.getString("media_id");
//数据同步,发送异步任务请求
if(null!=mediaId&&"!".equals(mediaId)){
    //获得 token
    String token=util.getTokenFromWx();
    String url=" https://qyapi.weixin.qq.com/cgi-bin/batch/replaceuser?
access_token="+token;
```

```
sendCVSData(mediaId ,url);
}
```

 **备注：**封装 CVS 文件发送方法 sendCVSData(String mediaId,String url)在 7.4.2 节中已有介绍，本节不再添加。示例代码中的“性别”字段在早期模板中有提供，目前模板已取消，但仍然支持填写，属于普通字段，1 表示男性，2 表示女性。

7.4.4 jobid 获取异步任务结果

异步任务在企业号中执行时间可能较长，除异步任务执行结束通过回调地址接收 (callback) 接收结果之外，还可以通过异步任务 ID (jobid) 主动获取异步任务执行结果，接口详细说明如下。

 **备注：**建议使用回调链接接收异步消息。

(1) Https 请求链接：

https://qyapi.weixin.qq.com/cgi-bin/batch/getresult?access_token=ACCESS_TOKEN&jobid=JOBID

(2) 数据请求方式。

GET 方式进行数据请求。

(3) 请求参数说明如表 7.20 所示。

表 7.20 jobid获取异步任务结果请求消息体参数说明

参数	是否必需	说明
access_token	是	调用接口凭证
jobid	是	异步任务ID，最大长度为64字节

(4) 返回结果及参数说明：

```
{
  "errcode": 0,
  "errmsg": "ok",
  "status": 1,
  "type": "replace_user",
  "total": 3,
  "percentage": 33,
  "remaintime": 1,
  "result": [{},{}]}
}
```

result 字段需要根据不同的类型展示不同的信息，且在任务完成后此字段才有效。当 type 为 replace_user 时，result 处理结果如下：

```
"result": [
```

```
{
  "action":1,
  "userid":"lisi",
  "errcode":0,
  "errmsg":"ok"
},
{
  "action":1,
  "userid":"zhangsan",
  "errcode":0,
  "errmsg":"ok"
}
]
```

当 type 为 replace_party 时，result 处理结果如下：

```
"result": [
  {
    "action":1,
    "partyid":1,
    "errcode":0,
    "errmsg":"ok"
  },
  {
    "action":4,
    "partyid":2,
    "errcode":0,
    "errmsg":"ok"
  }
]
```

参数详细说明如表 7.21 所示。

表 7.21 jobid 获取异步任务结果返回参数说明

参数	说明
errcode	返回码，0表示成功
errmsg	对返回码的文本描述内容
status	任务状态，整型 1表示任务开始； 2表示任务进行中； 3表示任务已完成
type	操作类型，字符串，目前分别有： replace_user（全量覆盖成员）； replace_party（全量覆盖部门）
total	任务运行总条数
percentage	目前运行百分比，当任务完成时为100
remaintime	预估剩余时间（单位：分钟），当任务完成时为0

续表

参数	说明
result	详细的处理结果，具体格式参考下面说明。当任务完成后此字段有效
action	操作类型（按位或） 当type为replace_user时： 1表示修改，2表示新增。 当type为replace_party时： 1新建部门，2更改部门名称，4移动部门，8修改部门排序
userid	成员ID，对应管理端的账号
partyid	部门ID

(5) 示例代码：

```
/**
 * 获得异步任务结果
 */
public JSONObject getResultByjobid(String jobid){
    //消息 JSON 格式
    //获得 Token
    String token=getTokenFromWx();
    try {
        CloseableHttpClient httpClient = HttpClients.createDefault();
        HttpPost httpPost= new HttpPost("https://qyapi.weixin.qq.com/cgi-bin/batch/"+
            "getresult?access_token="+token+"&jobid="+jobid);
        //创建请求处理类
        ResponseHandler<JSONObject> responseHandler = new ResponseHandler<
        JSONObject>() {
            public JSONObject handleResponse(
                final HttpResponse response) throws ClientProtocolException,
                IOException {
                int status = response.getStatusLine().getStatusCode();
                if (status >= 200 && status < 300) {
                    HttpEntity entity = response.getEntity();
                    if(null!=entity){
                        String result= EntityUtils.toString(entity);
                        //根据字符串生成 JSON 对象
                        JSONObject resultObj = JSONObject.fromObject(result);
                        return resultObj;
                    }else{
                        return null;
                    }
                } else {
                    throw new ClientProtocolException("Unexpected status:
                    " + status);
                }
            }
        }
```



```

    }
    };
    //返回的 JSON 对象
    JSONObject responseBody = httpClient.execute(httpPost, responseHandler);
    System.out.println(responseBody.toString());
    return responseBody;
} catch (Exception e) {
    e.printStackTrace();
    return null;
}
}

```

7.4.5 callback 接收异步任务通知

callback 参数为读者发送异步任务请求的回调链接，参数可以为空。用于异步任务完成后，通过 callback 推送事件给读者服务器。读者在配置回调链接时，可能出现“无效链接”的错误，主要原因在于链接的创建方式。链接的创建与回调模式链接创建相同，需要 GET 验证，POST 接收数据，示例代码如下。

(1) 在 web.xml 中创建服务连接 getJbidServlet:

```

<servlet>
    <servlet-name>getJbidServlet</servlet-name>
    <servlet-class>
        com.wx.servlet. GetJbidServlet
    </servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name> getJbidServlet </servlet-name>
    <url-pattern>/ getJbidServlet </url-pattern>
</servlet-mapping>

```

(2) 创建异步任务信息接收 servlet 类 GetJbidServlet.java:

```

public class GetJbidServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        //微信加密签名
        String signature = request.getParameter("msg_signature");
        //时间戳
        String timestamp = request.getParameter("timestamp");
        //随机数
        String nonce = request.getParameter("nonce");
        //随机字符串
        String echostr = request.getParameter("echostr");
        String sToken = WxUtil.RESP_MESSAGE_TOKEN;
    }
}

```

```

String sCorpID = WxUtil.RESP_MESSAGE_CORPID;
String sEncodingAESKey = WxUtil.RESP_MESSAGE_ENCODINGAESKEY;
try {
    WXBizMsgCrypt wxcpt = new WXBizMsgCrypt(sToken, sEncodingAESKey,
sCorpID);

    String sEchoStr; //需要返回的明文
    sEchoStr = wxcpt.VerifyURL(signature, timestamp, nonce, echostr);
    System.out.println("verifyurl echostr: " + sEchoStr);
    //验证 URL 成功, 将 sEchoStr 返回
    PrintWriter out = response.getWriter();
    out.write(sEchoStr);
    out.flush();
    out.close();
} catch (Exception e) {
    //验证 URL 失败, 错误原因请查看异常
    e.printStackTrace();
}
}

protected void doPost(HttpServletRequest request, HttpServletResponse
response)
    throws ServletException, IOException {
    //读取消息, 执行消息处理
    //微信加密签名
    String sReqMsgSig = request.getParameter("msg_signature");
    //时间戳
    String sReqTimeStamp = request.getParameter("timestamp");
    //随机数
    String sReqNonce = request.getParameter("nonce");
    String sToken = WxUtil.RESP_MESSAGE_TOKEN;
    String sCorpID = WxUtil.RESP_MESSAGE_CORPID;
    String sEncodingAESKey = WxUtil.RESP_MESSAGE_ENCODINGAESKEY;
    try {
        //POST 请求的密文数据
        ServletInputStream in = request.getInputStream();
        BufferedReader reader = new BufferedReader(new InputStreamReader
(in));


        String sReqData="";
        String itemStr=""; //作为输出字符串的临时串, 用于判断是否读取完毕
        while(null!=(itemStr=reader.readLine())){
            sReqData+=itemStr;
        }
        //对消息进行处理获得明文
        WXBizMsgCrypt wxcpt = new WXBizMsgCrypt(sToken, sEncodingAESKey, sCorpID);
        String sMsg = wxcpt.DecryptMsg(sReqMsgSig, sReqTimeStamp, sReqNonce,
sReqData);
        //输出解密后的文件
    }
}

```

```

        System.out.println("after decrypt msg: " + sMsg);
        //对解析出明文 XML 标签的内容进行处理
        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
        DocumentBuilder db = dbf.newDocumentBuilder();
        StringReader sr = new StringReader(sMsg);
        InputSource is = new InputSource(sr);
        Document document = db.parse(is);
        Element root = document.getDocumentElement();
        //获得消息内容
        NodeList nodelist_jobid = root.getElementsByTagName("jobid");
        String jobid = nodelist_jobid.item(0).getTextContent();
        System.out.println("获得的 jbid: "+jobid);
        //设置回复
        //回复人
        NodeList nodelist_fromUser = root.getElementsByTagName("FromUserName");
        String mycreate = nodelist_fromUser.item(0).getTextContent();
        //回复人
        //时间
        String time=new Date().getTime()+"";
        //应用 ID
        String AgentID="0";
        //消息类型
        String msg_type="text";
        //content="被动响应消息:"+content;
        String content="";
        //生成一个被动响应的消息
        TextMessage txtMsg= new TextMessage();
        txtMsg.setContent(content);//文字内容
        txtMsg.setCreateTime(Long.valueOf(time));//创建时间
        txtMsg.setFromUserName(sCorpID);//消息来源
        txtMsg.setMsgType("text");//消息类型
        txtMsg.setToUserName(mycreate);
        String sRespData=WxUtil.messageToXml(txtMsg);
        String sEncryptMsg = wxcpt.EncryptMsg(sRespData, time, sReqNonce);
        //输出
        PrintWriter out = response.getWriter();
        out.write(sEncryptMsg);
        out.flush();
        out.close();
    }catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

 备注：servlet 中的 messageToXml 等方法，可以参考“第 4 章 被动回调模式”。

(3) 修改异步任务请求 callback 参数，将 URL 设置为 getJbidServlet 服务地址。

7.5 标签管理

标签用于部门或成员创建特殊标记，可以通过标签配置不同的权限，如查看不同的企业号应用、向特定标签人员发送信息、通讯录不同的查看权限，等等。本节将演示如何通过接口对标签进行管理，讲解标签的创建、维护等操作。

7.5.1 创建标签

读者除了可以通过管理端创建标签外，还可以通过接口实现标签的创建，接口详细说明如下。

(1) Https 请求链接：

https://qyapi.weixin.qq.com/cgi-bin/tag/create?access_token=ACCESS_TOKEN

(2) 数据请求方式。

POST 方式进行数据请求。

(3) 消息请求结构体：

```
{
    "tagname": "UI",
    "tagid": id
}
```

(4) 参数说明详细说明如表 7.22 所示。

表 7.22 创建标签请求消息体参数说明

参数	是否必需	说明
access_token	是	调用接口凭证
tagname	是	标签名称，长度限制为32个字（汉字或英文字母），标签名不可与其他标签重名
tagid	否	标签ID，整型，指定此参数时新增的标签会生成对应的标签ID，不指定时则以目前最大的ID自增

(5) 权限说明。

创建的标签属于管理组，默认为加锁状态，加锁状态下只有本管理组才可以增删成员，解锁状态下，其他管理组也可以增删成员。

(6) 返回参数及说明：

```
{
    "errcode": 0,
    "errmsg": "created"
    "tagid": "1"}
}
```


返回参数中, errcode 值为 0 则表示执行成功, 其他结果为失败, tagid 为创建的标签 ID, 主要用于自动生成标签 ID 的读者。

(7) 示例代码:

```
/**
 * 创建标签
 * @param tagName 标签名
 * @return
 */
public String createTag(String tagName){
    String tagId="";
    //获取微信号
    String token=getTokenFromWx();
    try {
        CloseableHttpClient httpClient = HttpClients.createDefault();
        HttpPost httpPost= new HttpPost("https://qyapi.weixin.qq.com/cgi-bin/tag/create?access_token="+token);
        //发送 JSON 格式的数据
        StringEntity myEntity = new StringEntity("{\"tagname\":\""+tagName+"\"}");
        ContentType.create("text/plain", "UTF-8");
        httpPost.setEntity(myEntity);
        ResponseHandler<JSONObject> responseHandler = new
ResponseHandler<JSONObject>() {
            public JSONObject handleResponse(
                final HttpResponse response) throws
ClientProtocolException, IOException {
                int status = response.getStatusLine().getStatusCode();
                if (status >= 200 && status < 300) {
                    HttpEntity entity = response.getEntity();
                    if (null!=entity){
                        String result= EntityUtils.toString(entity);
                        //根据字符串生成 JSON 对象
                        JSONObject resultObj = JSONObject.fromObject
(result);

                        return resultObj;
                    }else{
                        return null;
                    }
                } else {
                    throw new ClientProtocolException("Unexpected status:
"+ status);
                }
            }
        };
    } catch (Exception e) {
        //返回的 JSON 对象
    }
}
```

```

        JSONObject responseBody = httpClient.execute(httpPost, responseHandler);
        tagId= responseBody.get("tagid")+"";
    }catch (Exception e) {
        e.printStackTrace();
        return null;
    }
    return tagId;
}

```

7.5.2 新增标签成员

职位调整、新员工入职、部门划分等情况引起的标签变化，可以通过接口实现成员、部门的新增，接口详细说明如下。

(1) Https 请求链接：

https://qyapi.weixin.qq.com/cgi-bin/tag/addtagusers?access_token=ACCESS_TOKEN

(2) 数据请求方式。

POST 方式进行数据请求。

(3) 消息请求结构体：

```

{
    "tagid": "1",
    "userlist": [ "user1", "user2"],
    "partylist": [4]
}

```

(4) 参数说明。详细说明如表 7.23 所示。

表 7.23 新增标签成员、部门请求消息体参数说明

参数	是否必需	说明
access token	是	调用接口凭证
tagid	是	标签ID
userlist	否	企业成员ID列表，注意：userlist、partylist不能同时为空，单次请求长度不能超过1000个字符
partylist	否	企业部门ID列表，注意：userlist、partylist不能同时为空，单次请求长度不能超过100个字符

(5) 权限说明。

当前管理组创建的标签或未加锁的标签，且成员属于当前管理组管辖范围内。

(6) 返回参数及说明。

结果信息分为三种形式：全部添加成功结果、部分成员或部门添加失败结果、全部添加失败结果。

当全部添加成功时，请求结果如下：

```

{
    "errcode": 0,

```

```
"errmsg": "ok"
}
```

当部分成员或部门添加失败时，请求结果如下：

```
{
  "errcode": 0,
  "errmsg": "错误消息",
  "invalidlist": "usr1|usr2|usr",
  "invalidparty": [2,4]
}
```



备注：invalidlist 中为添加失败的成员 ID，多个成员使用“|”分隔。invalidparty 为添加失败的部门，多个部门之间使用逗号分隔。

当全部添加失败时，请求结果如下：

```
{
  "errcode": 40070,
  "errmsg": "all list invalid "
}
```

(7) 示例代码：

```
/**
 * 增加标签人员、部门，部门和人员不能同时为空
 * @param tagId 标签 ID
 * @param userList 用户列表
 * @param partyList 部门列表
 * @return
 */
public boolean addInfoToTag(String tagId,List<String> userList,
List<String> partyList){
    boolean flag=true;
    //获取微信号
    String token=getTokenFromWx();
    //添加到标签的人员
    String userStr="";
    if(null!=userList){
        for (int i = 0; i < userList.size(); i++) {
            userStr+="\""+userList.get(i)+"\"";
            if(i!=userList.size()-1){
                userStr+=",";
            }
        }
    }
    if(!"".equals(userStr.trim())){
        userStr=",\"userlist\":[" +userStr+"]";
    }
}
```

```

    }
}
//添加到标签的部门
String departStr="";
if(null!=partyList){
    for (int i = 0; i < partyList.size(); i++) {
        departStr+=partyList.get(i);
        if(i!=partyList.size()-1){
            departStr+=",";
        }
    }
    if(!"".equals(userStr.trim())){
        departStr=",\"partylist\":[ "+departStr+"]";
    }
}
try {
    CloseableHttpClient httpClient = HttpClients.createDefault();
    HttpPost httpPost= new HttpPost("https://qyapi.weixin.qq.com/cgi-bin/tag/addtagusers?access_token="+token);
    //发送 JSON 格式的数据
    StringEntity myEntity = new StringEntity("{\"tagid\":\"1\\\""+userStr+departStr+"}");
    ContentType.create("text/plain", "UTF-8");
    httpPost.setEntity(myEntity);
    ResponseHandler<JSONObject> responseHandler =
        new ResponseHandler<JSONObject>() {
        public JSONObject handleResponse(
            final HttpResponse response) throws
ClientProtocolException, IOException {
            int status = response.getStatusLine().getStatusCode();
            if (status >= 200 && status < 300) {
                HttpEntity entity = response.getEntity();
                if(null!=entity){
                    String result= EntityUtils.toString(entity);
                    //根据字符串生成 JSON 对象
                    JSONObject resultObj = JSONObject.fromObject
(result);
                    return resultObj;
                }else{
                    return null;
                }
            } else {
                throw new ClientProtocolException("Unexpected status:
"+ status);
            }
        }
    }
}

```



```
};
//返回的JSON对象
JSONObject responseBody = httpclient.execute(httpPost,
responseHandler);
String errcode=""+ responseBody.get("errcode");
if("0".equals(errcode)){
    flag=true;
}else{
    flag=false;
}
}catch (Exception e) {
    e.printStackTrace();
    return false;
}
return flag;
}
```

7.5.3 删除标签成员

员工离职、部门合并等情况引起的标签变化，可以通过接口实现成员、部门的删除，接口详细说明如下。

(1) Https 请求链接:

https://qyapi.weixin.qq.com/cgi-bin/tag/deltagusers?access_token=ACCESS_TOKEN

(2) 数据请求方式。

POST 方式进行数据请求。

(3) 消息请求结构体:

```
{
    "tagid": "1",
    "userlist": [ "user1", "user2"],
    "partylist": [2,4]
}
```

(4) 参数说明。详细说明如表 7.24 所示。

表 7.24 删除标签成员、部门请求消息体参数说明


参数	是否必需	说明
access_token	是	调用接口凭证
tagid	是	标签ID
userlist	不能同时为空	企业成员ID列表
partylist		企业部门ID列表

(5) 权限说明。

当前管理组创建的标签或未加锁的标签，且成员属于当前管理组管辖范围内。

(6) 返回结果。

结果信息分为三种形式：全部删除成功结果、部分成员或部门删除失败结果、全部删除失败结果。

 **备注：**返回结果与 7.5.2 节新增标签成员返回结果相同，读者可以参考 7.5.2 节内容。

(7) 示例代码：

```
/**
 * 删除标签人员、部门，部门和人员不能同时为空
 * @param tagId 标签 ID
 * @param userList 用户列表
 * @param partyList 部门列表
 * @return
 */
public boolean deleteInfoFromTag(String tagId, List<String> userList,
List<String> partyList) {
    boolean flag=true;
    //获取微信号
    String token=getTokenFromWx();
    //删除标签的人员
    String userStr="";
    if(null!=userList){
        for (int i = 0; i < userList.size(); i++) {
            userStr+="\""+userList.get(i)+"\"";
            if(i!=userList.size()-1){
                userStr+=",";
            }
        }
        if(!"".equals(userStr.trim())){
            userStr=",\"userlist\":[" +userStr+"]";
        }
    }
    //删除标签的部门
    String departStr="";
    if(null!=partyList){
        for (int i = 0; i < partyList.size(); i++) {
            departStr+=partyList.get(i);
            if(i!=partyList.size()-1){
                departStr+=",";
            }
        }
        if(!"".equals(userStr.trim())){
            departStr=",\"partylist\":[" +departStr+"]";
        }
    }
    try {
```

```

        CloseableHttpClient httpClient = HttpClients.createDefault();
        HttpPost httpPost= new HttpPost("https://qyapi.weixin.qq.com/cgi-bin/tag/deltagusers?access_token="+token);
        //发送 JSON 格式的数据
        StringEntity myEntity = new StringEntity("{\"tagid\":\"1\""+userStr+departStr+"}",
            ContentType.create("text/plain", "UTF-8"));
        httpPost.setEntity(myEntity);
        //创建响应处理类
        ResponseHandler<JSONObject> responseHandler = new ResponseHandler<JSONObject>() {
            public JSONObject handleResponse(
                final HttpResponse response) throws ClientProtocolException,
                IOException {
                int status = response.getStatusLine().getStatusCode();
                if (status >= 200 && status < 300) {
                    HttpEntity entity = response.getEntity();
                    if(null!=entity){
                        String result= EntityUtils.toString(entity);
                        //根据字符串生成 JSON 对象
                        JSONObject resultObj = JSONObject.fromObject(result);
                        return resultObj;
                    }else{
                        return null;
                    }
                } else {
                    throw new ClientProtocolException("Unexpected status:
" + status);
                }
            }
        };
        //返回的 JSON 对象
        JSONObject responseBody = httpClient.execute(httpPost,
responseHandler);
        String errcode="" + responseBody.get("errcode");
        if("0".equals(errcode)){
            flag=true;
        }else{
            flag=false;
        }
    }catch (Exception e) {
        e.printStackTrace();
        return false;
    }
    return flag;
}

```

7.5.4 获取标签成员

通过标签获得当前标签下成员、部门列表，接口详细说明如下。

(1) Https 请求链接：

https://qyapi.weixin.qq.com/cgi-bin/tag/get?access_token=ACCES_TOKEN&tagid=TAGID

(2) 数据请求方式。

GET 方式进行数据请求。

(3) 请求参数说明。详细说明如表 7.25 所示。

表 7.25 获取标签成员、部门请求消息体参数说明

参数	是否必需	说明
access_token	是	调用接口凭证
tagid	是	标签ID

(4) 权限说明。

返回列表中仅能查看管理组管辖内的成员。

(5) 返回结果：

```
{
  "errcode": 0,
  "errmsg": "ok",
  "userlist": [
    {
      "userid": "zhangsan",
      "name": "李四"
    }
  ],
  "partylist": [2]
}
```

errcode 为执行结果，0 表示成功，1 表示失败，参数详细说明如表 7.26 所示。

表 7.26 获取部门列表响应消息参数说明

参数	说明
errcode	返回码
errmsg	对返回码的文本描述内容
userlist	成员列表
userid	成员UserID
name	成员姓名
partylist	部门列表

7.5.5 删除标签

读者通过接口能够实现标签的删除，接口详细说明如下。

(1) Https 请求链接:

https://qyapi.weixin.qq.com/cgi-bin/tag/delete?access_token=ACCESS_TOKEN&tagid=TID

(2) 数据请求方式:

GET 方式进行数据请求。

(3) 参数说明详细说明如表 7.27 所示。

表 7.27 创建标签请求消息体参数说明

参数	是否必需	说明
access_token	是	调用接口凭证
tagid	否	标签ID

(4) 权限说明。

管理组必须是指定标签的创建者，并且标签的成员列表为空。

(5) 返回参数及说明:

```
{"errcode": 0, "errmsg": "deleted"}
```

7.6 案例：企业通讯录异步维护

为了更好地掌握、理解本章内容，本节将通过案例——企业通讯录异步维护，学习企业通讯录的维护。企业通讯录维护可以通过定时执行实现企业部门人员的定期同步，保证入职、离职、调岗等人员的关注及权利，并同时实现以下需求：

- 同步人员、部门时，原有关注人员不能“被取消”关注。
- 按照手机号、微信号进行同步（员工必须填写手机号或微信号）。
- 人力资源、管理层等特殊部门除外，只需手机号、微信号、邮箱三者不同时为空。

通过注释与代码的方式，为读者演示企业通讯录的维护，示例代码如下：

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import net.sf.json.JSONObject;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.ResponseHandler;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.ContentType;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;
public class TestMain {
```

```

public static void main(String[] args) {
    //同步部门信息
    TestMain aa = new TestMain();
    aa.startDepartmentToWx();
}
public static String WEBURL="http://www.XXXXX.com/XXX";//外网域名
public static String RESP_MESSAGE_TOKEN="XXX";//回调模式配置的 token
public static String RESP_MESSAGE_ENCODINGAESKEY="XXXXX";//AES 密钥
public static String MASTER_PEOPLE="";
/**
 * 向微信同步数据
 * 0 成功 1 失败
 */
public String startDepartmentToWx() {
    //不需要严格验证的部门
    String partExtSql=" (partyName like '%管理层%' or partyName like '%
    人力资源%') and partyType='party' ";
    List<WxParty> highPartyList = new ArrayList<WxParty>();
    //----数据库获取省略
    List<WxContacts> contactsList = new ArrayList<WxContacts>();
    //----数据库获取人员省略
    //获取全部的人员, 将部门号设置为 1, 获得 cvs_contacts_temp, 用于保证已关注人员
    不会被取消
    StringBuffer cvs_contacts_temp=new StringBuffer("姓名,账号,微信号,手
    机号,邮箱,性别,所在部门,职位").append("\n");
    //部门存在真实的 cvs_contacts
    StringBuffer cvs_contacts=new StringBuffer("姓名,账号,微信号,手机号,邮
    箱,性别,所在部门,职位").append("\n");
    for (int i = 0; i < contactsList.size(); i++) {
        WxContacts contact = contactsList.get(i);
        if(true==checkParty(contact.getDepartment(),highPartyList)){
            //如果是特殊部门人员, 只需验证三个不能同时为空
            if("").equals(contact.getWeixinId())&&"".equals(contact.getMobile())
            &&"".equals(contact.getEmail())){
                continue;
            }
            if(null==contact.getWeixinId()||"".equals(contact.
            getWeixinId())){
                contact.setWeixinId("");
            }
            if(null==contact.getEmail()||"".equals(contact.getEmail())){
                contact.setEmail("");
            }
            if(null==contact.getPosition()||"".equals(contact.getPosition())){
                contact.setPosition("");
            }
        }
    }
}

```

```

        if(null==contact.getMobile()||"".equals(contact.getMobile())){
            contact.setMobile("");
        }
    }else{
        //性别数据库无字段，暂时默认为1
        if(null==contact.getWeixinId()||"".equals(contact.getWeixinId())){
            contact.setWeixinId("");
        }
        if(null==contact.getMobile()||"".equals(contact.getMobile())){
            if(null==contact.getWeixinId()||"".equals(contact.
getWeixinId())){
                //如果手机号为空，微信号也为空，则不同步
                continue;
            }else{
                //如果手机号为空，微信号不为空，则同步
                contact.setMobile("");
            }
        }
        if(null==contact.getEmail()||"".equals(contact.getEmail())){
            contact.setEmail("");
        }
        if(null==contact.getPosition()||"".equals(contact.getPosition())){
            contact.setPosition("");
        }
        if("".equals(contact.getWeixinId())&&"".equals(contact.
getMobile())
        &&"".equals(contact.getEmail())){
            continue;
        }
    }
    cvs_contacts_temp.append(contact.getUserName()+" "+contact.
getUserId()
    +" "+contact.getWeixinId()+" "+contact.getMobile()+" "+contact.getEmail(
)+" "+"1"+" "+"1"+" "+contact.getPosition()).append("\n");
    cvs_contacts.append(contact.getUserName()+" "+contact.
getUserId()+" "+
    contact.getWeixinId()+" "+contact.getMobile()+" "+contact.getEmail()+" "
+"1"+" "+contact.getDepartment()+" "+contact.getPosition()).append("\n");
}
System.out.println(cvs_contacts.toString());
//上传 CSV 文件到腾讯服务器
WxUtil util=new WxUtil();
String cvs_contacts_temp_mediaId="";
String cvs_contacts_mediaId="";
try {
    JSONObject json = util.sendCVSFile(cvs_contacts_temp.toString());

```



```

        cvs_contacts_temp_mediaId=json.getString("media_id");
        JSONObject json2 = util.sendCVSFile(cvs_contacts.toString());
        cvs_contacts_mediaId=json2.getString("media_id");
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    //数据同步
    if(null!=cvs_contacts_temp_mediaId&&!"".equals(cvs_contacts_temp_
mediaId)){
        //获得 token
        String token=WxUtil.getToken();
        String url="https://qyapi.weixin.qq.com/cgi-bin/batch/
replaceuser?access_token="
            +token;
        sendCVSData(cvs_contacts_temp_mediaId,util,url);
    }
    //从数据库获取所有部门数据生成 CSV 的数据
    StringBuffer content = new StringBuffer("部门名称,部门 ID,父部门 ID,排
序").append("\n");
    List<WxParty> partyList = new ArrayList<WxParty>();
    //-----数据库获取数据省略
    for (int i = 0; i < partyList.size(); i++) {
        WxParty party = partyList.get(i);
        if(null==party.getParentId()){
            party.setParentId((long)0);
        }
        //拼接字符串
        content.append(party.getPartyName()+","+party.getPartyId()+", "
            +party.getParentId()+", "+i).append("\n");
    }
    System.out.println(content.toString());
    //上传 CSV 文件到腾讯服务器
    String mediaId="";
    try {
        JSONObject json = util.sendCVSFile(content.toString());
        System.out.println(json.toString());
        mediaId=json.getString("media_id");
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    //数据同步
    if(null!=mediaId&&!"".equals(mediaId)){
        //获得 token
        String token=WxUtil.getToken();

```



```

        String url="https://qyapi.weixin.qq.com/cgi-bin/batch/
replaceparty?access_token="
            +token;
        sendCVSData(mediaId,util,url);
    }
    //数据同步
    if(null!=cvs_contacts_mediaId&&"".equals(cvs_contacts_mediaId)){
        //获得 token
        String token=WxUtil.getToken();
        String
url="https://qyapi.weixin.qq.com/cgi-bin/batch/replaceuser?access_token="
            +token;
        sendCVSData(cvs_contacts_mediaId,util,url);
    }
    System.out.println("同步成功");
    return "1";
}
/**
 * 向企业号发送 CVS 文件
 * @param mediaId 媒体 ID
 * @param util 工具类
 * @param url 企业号不同接口链接
 * @return
 */
public boolean sendCVSData(String mediaId,WxUtil util,String url){
    String jsonContext="{ "+
        "\"media_id\": \""+mediaId+"\", "+
        "\"callback\": "+
        "{ "+
            "\"url\": \""+WEBURL+"/addressBookGetJbidServlet"+"\", "+
            "\"token\": \""+RESP_MESSAGE_TOKEN+"\", "+
            "\"encodingaeskey\": \""+RESP_MESSAGE_ENCODINGAESKEY+"\" "+
        "}"+"
    }";
    //发送消息
    //消息 JSON 格式
    boolean flag=false;
    try {
        CloseableHttpClient httpClient = HttpClients.createDefault();
        HttpPost httpPost= new HttpPost(url);
        //发送 JSON 格式的数据
        StringEntity myEntity = new StringEntity(jsonContext,
            ContentType.create("text/plain", "UTF-8"));
        httpPost.setEntity(myEntity);
        // Create a custom response handler
        ResponseHandler<JSONObject> responseHandler = new ResponseHandler

```

```

<JSONObject>() {
    public JSONObject handleResponse(
        final HttpResponse response) throws ClientProtocolException,
        IOException {
        int status = response.getStatusLine().getStatusCode();
        if (status >= 200 && status < 300) {
            HttpEntity entity = response.getEntity();
            if (null != entity) {
                String result = EntityUtils.toString(entity);
                //根据字符串生成 JSON 对象
                JSONObject resultObj = JSONObject.fromObject(
                    result);
                return resultObj;
            } else {
                return null;
            }
        } else {
            throw new ClientProtocolException("Unexpected response
status: " + status);
        }
    }
    //返回的 JSON 对象
    JSONObject responseBody = httpClient.execute(httpPost,
responseHandler);
    System.out.println(responseBody);
    int result = (Integer) responseBody.get("errcode");
    if (0 == result) {
        flag = true;
    } else {
        flag = false;
    }
    httpClient.close();
} catch (Exception e) {
    e.printStackTrace();
    return false;
}
return true;
}

/**
 * 判断当前部门是不是特殊部门
 * @param curPartyId
 * @param list
 * @return
 */

```

```
public boolean checkParty(Long curPartyId, List<WxParty> list){
    for (int i = 0; i < list.size(); i++) {
        if(curPartyId.longValue()==list.get(i).getPartyId().longValue()){
            return true;
        }
    }
    return false;
}
```




备注：本案例通过三步实现企业部门、人员的同步。首先将企业号已关注人员部门置为 1（根部门），然后进行企业号部门的同步，最后将所有人员的部门进行同步更新。上传 CVS 文件的方法 sendCVSFile 可以参照 7.4.1 节。

数据安全访问策略

服务器网络由内到外分为内网、DMZ 区和外网。内网为企业内部使用的局域网，外网多指因特网，而 DMZ 区为内网与外网之间的服务器缓存区。内网数据通过 DMZ 区传递至外网，而外网则通过 DMZ 区请求所需要的数据。

微信企业号开发同样需要对数据进行安全防护，企业号能够与企业 OA、IM 和 HR 等系统进行对接，实现内部实时资讯、移动即时通信和微信考勤等。企业 OA、IM 和 HR 等系统均部署于内网系统，企业号服务则需要部署于 DMZ 区，实现与因特网之间的数据交互，防止黑客入侵、员工误转发等。为了让读者更好地理解企业号中如何进行数据安全访问，本章将为读者介绍以下数据安全策略。

- 微信 OAuth 2.0 身份验证：学习如何进行微信身份验证，保证使用人身份。
- 浏览器类型安全访问：掌握浏览器的判断方式，只需在微信内置浏览器中访问。
- 全局验证码变量：掌握如何利用全局验证码保证本次数据访问的唯一有效性。
- 页面有效期访问：如何确保数据访问时间。
- QPID 数据提取：如何通过 AMQP 协议进行数据访问。
- 代理服务器：掌握如何利用中间服务器以及数据流的形式获取数据、解析数据。
- 数据安全访问策略案例：通过 DMZ 服务器获取内网图片，使读者更好地理解数据安全访问的方式，更好地掌握如何进行内网数据安全交互。

 备注：DMZ 区，又称“隔离区”、“非军事化隔离区”、“危险管理区”，属于内网中的一个特殊区域，是内外网数据分离所必需的服务缓存区。

8.1 OAuth 2.0 身份验证

OAuth 2.0 验证用于企业号开发 URL 链接获取成员身份信息的接口，成员通过点击具有身份验证的链接访问读者服务时，读者服务将获得当前访问的成员身份信息。注意，此处 URL 链接的域名必须完全匹配企业应用设置项中的“可信域名”，否则跳转时会提示 `redirect_uri` 参数错误。如果读者 `redirect_uri` 有端口号，如图 8.1 所示，则 URL 链接必须加上端口号，如 `http://www.xxxx.com:6503/WX_DEMO/coreServlet`。

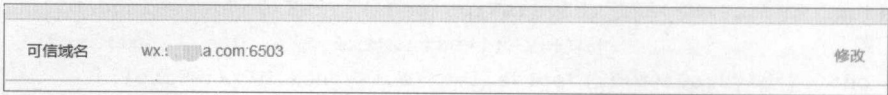


图 8.1 具有端口号的可信域名

OAuth 2.0 身份验证主要分为两步：

- 01 将读者 URL 链接处理成特殊的 OAuth 2.0 链接，用于从读者服务获得 code；
- 02 利用 code 向企业号获取成员身份信息。

本节将向读者演示如何通过特定的 OAuth 2.0 链接获得成员信息，便于读者进行相应的操作。

8.1.1 获取 code

如果读者需要成员在跳转到网页时带上员工的身份信息，则需要构造如下链接：

`https://open.weixin.qq.com/connect/oauth2/authorize?appid=CORPID&redirect_uri=REDIRECT_URI&response_type=code&scope=SCOPE&state=STATE#wechat_redirect`

其中 redirect_uri 为读者服务页面，链接需要进行 urlencode 处理，如：

`http://www.muyunfei.com:6503/WX_DEMO/coreServlet`


经过 urlencode 处理后链接变为：

`http%3a%2f%2fwww.muyunfei.com%3a6503%2fWX_DEMO%2fcoreServlet`

其他参数详细说明如表 8.1 所示。

表 8.1 OAuth 2.0 链接参数

参数	是否必需	说明
appid	是	企业的Corpid
redirect_uri	是	授权后重定向的回调链接地址,请使用urlencode对链接进行处理
response_type	是	返回类型,此时固定为: code
scope	是	应用授权作用域,此时固定为: snsapi base
state	否	重定向后会带上state参数,企业可以填写a~z A~Z 0~9的参数值,长度不可超过128字节
#wechat_redirect	是	微信终端使用此参数判断是否需要带上身份信息

 备注：员工点击后，页面将跳转至 `redirect_uri?code=CODE&state=STATE`，读者可根据 code 参数获得员工的 userid。


JS 处理链接示例代码如下：

```
// 查看详情
function showoaDetail(msgId) {
```

```
//原始链接
var getUrl="http://www.muyunfei.com:6503/WX_DEMO/coreServlet.do?msgId=
"+msgId;
//链接处理
var changeurl=getUrl.replace(/[:]/g,"%3a").replace(/[\/]/g,"%2f").replace
(/[\?]/g,"%3f")
                .replace(/[=]/g,"%3d").replace(/[&]/g,"%26");
var tourl="https://open.weixin.qq.com/connect/oauth2/authorize?"
        +"appid=wxdl87a9lce6c62d27&redirect_uri="+changeurl
        +"&response_type=code&scope=snsapi_base&state=location#wechat_
redirect";
//链接跳转
location.href=tourl;
}
```

8.1.2 根据 code 获得成员信息

通过成员接口利用成员授权获取到的 code 便能够获得当前访问的成员，详细说明如下：



注意：同一成员每次授权带上的 code 并不相同，且 code 只能使用一次，5 分钟未被使用则自动过期。

- (1) Https 请求链接：
https://qyapi.weixin.qq.com/cgi-bin/user/getuserinfo?access_token=XX &code=CODE
- (2) 数据请求方式。
Get 方式进行数据请求。
- (3) 请求参数说明：详细说明如表 8.2 所示。

表 8.2 获取成员请求消息体参数说明

参数	是否必需	说明
access_token	是	调用接口凭证
code	是	通过成员授权获取到的code

- (4) 权限说明。
跳转链接的域名必须完全匹配可信域名。
- (5) 返回结果：

```
{
  "UserId":"USERID",
  "DeviceId":"DEVICEID"
}
```



备注: DeviceID 为手机设备号, 在微信安装时随机生成, 删除重装会改变, 升级不受影响。同一设备上不同的登录账号生成的 deviceid 也不相同。双号模式 (企业号+服务号) 下, 非企业号成员获得的人员信息为 OpenID, 而非 UserID。

(6) 示例代码:

```
/**
 * 根据 code 获得人员
 */
public String getUserIdByCode(String code){
    //获取请求票据
    String token=getTokenFromWx();
    try {
        CloseableHttpClient httpclient = HttpClients.createDefault();
        HttpPost httpPost= new HttpPost("https://qyapi.weixin.qq.com/cgi-bin/"
            +"user/getuserinfo?access_token="+token+"&code="+code);
        //创建自定义响应处理程序
        ResponseHandler<JSONObject> responseHandler =
            new ResponseHandler<JSONObject>() {
            public JSONObject handleResponse(
                final HttpResponse response) throws ClientProtocolException,
                IOException {
                int status = response.getStatusLine().getStatusCode();
                if (status >= 200 && status < 300) {
                    HttpEntity entity = response.getEntity();
                    if(null!=entity){
                        String result= EntityUtils.toString(entity);
                        //根据字符串生成 JSON 对象
                        JSONObject resultObj = JSONObject.fromObject
                            (result);
                        return resultObj;
                    }else{
                        return null;
                    }
                } else {
                    throw new ClientProtocolException("Unexpected status:
                        " + status);
                }
            }
        };
        //返回的 JSON 对象
        JSONObject responseBody = httpclient.execute(httpPost,
            responseHandler);
        if(null==responseBody.getString("UserId")){
            return null;
        }
    }
}
```

```

    }else{
        return responseBody.getString("UserId");
    }
} catch (Exception e) {
    //e.printStackTrace();
    return null;
}
}

```

8.2 浏览器类型安全访问

在第5章 JSAPI 模式中,已经了解到微信内置浏览器,针对浏览器的类型我们可以设置相应的安全策略——仅允许在微信内置浏览器中打开。

通过 `ServletActionContext.getRequest().getHeader("User-Agent")` 获得当前浏览器代理信息,各类型浏览器代理信息如下:

(1) IE 浏览器:

Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)

(2) Google 浏览器:

Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/45.0.2454.93 Safari/537.36

(3) 360 安全浏览器:

Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/45.0.2454.101 Safari/537.36

(4) UC 浏览器:

Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/50.0.2661.102 UBrowser/5.7.15319.202 Safari/537.36

(5) 手机 QQ 浏览器:

Mozilla/5.0 (Linux; Android 4.2.2; N1W Build/JDQ39) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/37.0.0.0 Mobile MQQBrowser/6.2 TBS/036558 Safari/537.36
V1_AND_SQ_6.5.0_390_YYB_D QQ/6.5.0.2835 NetType/WIFI WebP/0.3.0 Pixel/1080

(6) Android 微信内置浏览器:

Mozilla/5.0 (Linux; Android 4.2.2; N1W Build/JDQ39) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/37.0.0.0 Mobile MQQBrowser/6.2 TBS/036558 Safari/537.36
MicroMessenger/6.3.23.840 NetType/WIFI Language/zh_CN

(7) iPhone 微信内置浏览器:

Mozilla/5.0 (iPhone; CPU iPhone OS 9_3_5 like Mac OS X) AppleWebKit/601.1.46 (KHTML, like Gecko) Mobile/13G36 MicroMessenger/6.3.24 NetType/4G Language/zh_CN

通过对比多个浏览器的代理信息,可以发现一个关键信息“MicroMessenger”。通过“MicroMessenger”便能够区分请求的来源,从而保证信息只能在微信中打开,示例代码如下:


```

HttpServletRequest req = ServletActionContext.getRequest();
//识别微信浏览器
String userAgent=req.getHeader("User-Agent");//里面包含了设备类型
if(-1==userAgent.indexOf("MicroMessenger")){
    //如果不是微信浏览器,则跳转到安全页
    return "safePage";
}


```

JSP 页面中,防止外部浏览器打开方法的示例代码如下:

```

<%
//识别微信浏览器
String userAgent=request.getHeader("User-Agent");//里面包含了设备类型
if(-1==userAgent.indexOf("MicroMessenger")){
    //如果不是微信浏览器,则跳转到安全页
    request.getRequestDispatcher("noRightPage.jsp").forward(request,
response);
}
%>

```

 **备注:** 在微信中,可以通过 `userAgent.indexOf("iPhone")` 来区分是 Android 手机还是 iPhone 手机,示例代码如下:


```

HttpServletRequest req = ServletActionContext.getRequest();
String userAgent=req.getHeader("User-Agent");//里面包含了设备类型
if(-1!=userAgent.indexOf("iPhone")){
//-----如果是苹果手机-----//
//此方法需要浏览器自己能够打开,iOS 可以,但是微信 Android 版内置浏览器不支持
}else{
//如果非苹果手机,则自己处理文档
}

```

8.3 全局验证码变量

全局验证码变量 `regesiterCurMaxCode`,为当前页面所操作的库表最大主键,所有新增数据的主键只能小于等于 `regesiterCurMaxCode`,且数据库中不能存在当前主键。此变量主要用于解决链接暴露而引起的数据新增问题。

 **备注:** 可能认为使用 filter 过滤器过滤未登录即可,但是微信开发中无法使用过滤登录状态。如果使用,那么所有的链接都将被屏蔽,除非在请求中带入账号信息(带入登录信息则不必进行过滤)。

首先,定义全局变量 `regesiterCurMaxCode`,如下所示:

```
//操作库表的最大的主键
public static long regesiterCurMaxCode=0;
```

在成员进入新增页面时，初始化页面数据：

```
//生成 checkLogId 主键——ID 作为验证码
long checkLogId = SequenceUtil.genEntitySequenceNo(WxCheckLog.class);
WxUtil.regesiterCurMaxCode=checkLogId;
req.setAttribute("checkInfo", checkLogId);
```

注意：checkCode 的命名尽量使用无意义的名字，用于数据混淆。

成员点击新增按钮时，再将 checkCode 传递回后台用于校验。验证主要分两种：（1）校验是否超过全局变量，超过则为恶意刷新；（2）判断数据库中是否已经存在，存在则表示恶意刷新数据。示例代码如下：


```
//验证 checkID 是否在范围内，不在范围内则为恶意刷新
long checkId=0;
if(null==req.getParameter("checkCode")||"".equals(req.getParameter("checkCode"))||Long.valueOf(req.getParameter("checkCode"))>WxUtil.regesiterCurMaxCode){
    //跳转至安全页面
    return "safePage";
}
checkId=Long.valueOf(req.getParameter("checkCode"));
//判断是否在数据库中，若存在则为恶意刷新
WxCheckLog isEixtData = wxCheckLogServiceImpl.retrieveById(checkId);
if(null!=isEixtData){
    //跳转至安全页面
    req.setAttribute("curUser", isEixtData.getUserId());
    return "safePage";
}
```

备注：全局验证码变量是安全策略中的一种策略，有两种问题：新增页面初始化即产生数据库主键会造成主键空缺；小于验证且数据库中不存在的空缺 ID，仍然可以被用于新增（能够成功命中空缺 ID 的几率较低）。

8.4 页面有效期访问

页面有效期访问，主要实现成员在进入页面后仅能在一段时间内进行操作，对于超过规定时间的成员将禁止操作。用于防止成员信息泄露或超时处理等。页面有效期访问分为前台 JS 校验和后台时间校验，前台校验能够解决业务上的效果实现，而后台校验主要用于防止系统漏洞，增加系统安全性，应用场景如下：

场景一，成员薪资水平对于公司来说较为保密，减少员工之间不必要的薪资泄露是企业稳

 备注: setInterval() 方法是不停地调用函数,直到 clearInterval() 被调用或窗口被关闭,而 setTimeout() 是一定时间后执行“一次”函数。

8.4.2 事件校验

事件校验,分为 JS 事件时间校验和后台 session 时间校验。当前台点击某个按钮时,能够通过 JS 获取本机时间进行校验。为了防止成员修改本机时间创建程序漏洞,因而增加后台时间校验,通过校验服务器时间来增加系统的安全性。

(1) JS 校验时间。

当员工进行微信考勤,点击“考勤打卡”时,可校验员工是否长时间保持当前页面定位。如果超过有效时间,则需要员工重新获取定位后,方可以进行考勤。示例代码如下:

```
<html>
<head>
  <title>JS 时间校验</title>
</head>
<body>
  当前 GPS 位置: <span id="curTime"></span><br/>
  经度:101.60156187500003<br/>
  纬度:35.603717439502084<br/>
  <div onclick="regesiter()" style="background-color:red;width:100px">考
勤打卡</div>
</body>
<script>
  var flag=60; //有效值,单位秒
  var beginTime=new Date();//初始化开始时间
  function regesiter(){
    //获取单击事件的当前时间
    var endTime=new Date();
    var temp=endTime.getTime()-beginTime.getTime();
    if(temp>flag*1000){
      alert("页面超时失效,需要重新刷新,是否刷新?");
      return ;
    }else{
      //继续运行
      alert("继续运行");
    }
  }
</script>
</html>
```

(2) 后台时间校验。

后台校验主要用于防止成员通过修改本机时间,从而跳过 JS 时间验证的问题。在页面初始化时存入当前时间,示例代码如下:


```
request.getSession().setAttribute("bgTime", new Date());
```

当点击“考勤打卡”后，通过 JS 校验进入后台进行时间校验，示例代码如下：

```
long interval=60L;//阈值
Date bgTime = (Date) request.getSession().getAttribute("bgTime");
Date edTime = new Date();
if((edTime.getTime()-bgTime.getTime())>interval*1000L){
    //重新设置，session
    PrintWriter out = response.getWriter();
    out.print("页面超时");
    out.flush();
    out.close();
}
```



备注：后台增加时间校验后，还需要进行 JS 校验是为了减少服务端压力。

8.5 QPID 消息队列

QPID，是 Apache 提供的一个消息服务器，是 AMQP 协议（Advanced Message Queuing Protocol，高级消息队列协议）的一种实现，通过“消息队列”的方式，实现内外网数据的提前，DMZ 区服务将消息放入队列中，再由 DMZ 区服务主动获取结果队列消息，内网区同样实现主动获取请求队列信息，主动存放结果队列信息，对 QPID 感兴趣的读者可以去官网查看详细信息，本节抛砖引玉，简单介绍 QPID 的使用。



备注：QPID 服务稳定性较差，消息发送、接收需要读者进行加锁处理，以防止消息并发而引起的消息混乱或无法读取问题。

8.5.1 QPID 消息 Hello World

QPID 分为服务端和客户端，读者可以从官网下载，下载地址如下：

<https://qpid.apache.org/download.html>

其中 qpid-broker-0.32-bin.tar.gz 为服务端（以 0.32 版本为例），qpid-client-0.32-bin.tar.gz 为客户端，读者需根据自身情况下载对应的开发语言。

在开发 QPID 消息队列时，读者需要首先确保 QPID 服务端开启。点击“qpid-broker-0.32-bin\qpid-broker\0.32\bin\qpid-server.bat”启动 QPID 服务端，如图 8.2 所示。

备注：如果出现 Unsupported major.minor version 51.0 异常，则可能是下载的 qpid jar 包的编译版本太高。一种方法是对反编译 jar 包重新编译，另一种方法是提高 JDK 版本。例如 apache-qpid-jms-0.5.0 是由 JDK 1.7 版本编译，需要至少 JDK 1.7 以上版本来运行。

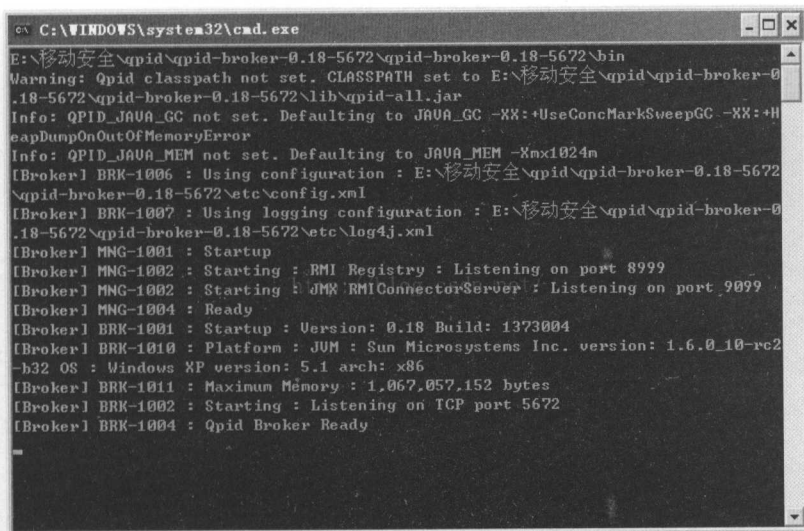


图 8.2 启动 QPID 服务端

QPID 启动后, 控制台 (HTTP 服务端) 默认端口为 8080, 服务端 (AMQP 服务端) 为 5672, 可通过如下 cmd 命令进行修改:

```
$ ./qpidd-server -prop "qpidd.amqp_port=10000" -prop "qpidd.http_port=10001"
```

接下来, 在准备工作完成后, 学习程序开发经典示例 “HELLO WORLD”, 在同一目录下创建 QpidHelloWorld.java 和 hello.properties 两个文件, 配置文件 hello.properties 示例代码如下:

```
java.naming.factory.initial =
org.apache.qpid.jndi.PropertiesFileInitialContextFactory
# connectionfactory.[jndiname] = [ConnectionFactory]
connectionfactory.qpidConnectionFactory =
amqp://guest:guest@test/?brokerlist='tcp://172.20.37.77:5672'
# Register an AMQP destination in JNDI
# destination.[jniName] = [Address Format]
destination.topicExchange = amq.topic
```

QpidHelloWorld.java 示例代码如下:

```
package myf.caption8.qpid;
import java.io.InputStream;
import java.util.Properties;
import javax.jms.Connection;
import javax.jms.ConnectionFactory;
import javax.jms.Destination;
import javax.jms.MessageConsumer;
import javax.jms.MessageProducer;
import javax.jms.ObjectMessage;
import javax.jms.Session;
import javax.jms.TextMessage;
```

```

import javax.naming.Context;
import javax.naming.InitialContext;
public class QpidHelloWorld {
    public static void main(String[] args) {
        QpidHelloWorld hello = new QpidHelloWorld();
        hello.runTest();
    }
    private void runTest() {
        try{
            //获得配置文件
            InputStream resourceAsStream = this.getClass().getResourceAsStream
("hello.properties");
            Properties properties = new Properties();
            properties.load(resourceAsStream);
            //使用配置文件创建JNDI的上下文,这里指的是PropertiesFileInitialContextFactory
            Context context = new InitialContext(properties);
            //从JNDI中获取链接工厂 qpidConnectionFactory
            ConnectionFactory connectionFactory = (ConnectionFactory)
context.lookup("qpidConnectionFactory");
            //使用链接工厂创建链接 amqp://guest:guest@test/?brokerlist= 'tcp:
//localhost:5672'
            //格式如下 amqp://[<user>:<pass>@][<clientid>]<virtualhost>
[?<option>='<value>' [&<option>='<value>']]
            //brokerlist 的格式 如下 brokerlist=<transport>://<host>[:<port>]
(?<param>='<value>') (&<param>='<value>')*
            Connection connection = connectionFactory.createConnection();
            connection.start();
            //在链接内创建会话
            Session session = connection.createSession(false, Session.AUTO_
ACKNOWLEDGE);
            //从JNDI中获取目的地址, 这里的目的是 amq.topic
            //amq.topic 类似于 jms 的 发布/订阅模式
            Destination destination = (Destination) context.lookup
("topicExchange");
            //从会话中产生生产者与消费者
            MessageProducer messageProducer = session.createProducer
(destination);
            MessageConsumer messageConsumer = session.createConsumer
(destination);
            //产生文本消息
            TextMessage message = session.createTextMessage("Hello world!");
            //产生自定义实体消息
            User user1 = new User("muyunfei", "123456");
            ObjectMessage objectMessage = session.createObjectMessage();
            objectMessage.setObject(user1);
            //发送文本消息

```

```

        messageProducer.send(message);
        //messageProducer.send(objectMessage);
        //接收消息
        message = (TextMessage)messageConsumer.receive();
        System.out.println(message.getText());
        //接收自定义实体消息
        ObjectMessage mes= (ObjectMessage)messageConsumer.receive();
        System.out.println(((User)mes.getObject()).getUserName());
        //关闭资源链接
        connection.close();
        context.close();
    }
    catch (Exception exp) {
        exp.printStackTrace();
    }
}
}

```

 **备注：**发送文件消息使用 `javax.jms.TextMessage` 对象。如果读者需要发送自定义实体，则可以使用 `javax.jms.ObjectMessage` 实体，示例代码如下：

```

User user1 = new User("muyunfei","123456");
ObjectMessage objectMessage = session.createObjectMessage();
objectMessage.setObject(user1);
messageProducer.send(objectMessage);

```

所需 jar 包下载地址：<http://download.csdn.net/detail/myfmyfmyfmyf/9244513>。

8.5.2 QPID 发送 MAP 消息

至此，我们已经学会了基本的 QPID 使用，本节将介绍多个客户端（`MapSender` 信息发送人和 `MapReceiver` 信息接收人）如何借助 QPID 消息队列完成消息发送，首先建立如图 8.3 所示工程。

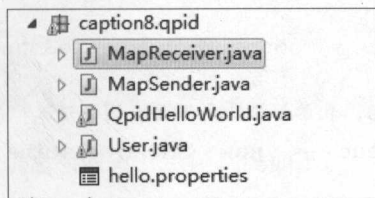


图 8.3 QPID 工程目录

由客户端 `MapSender` 发送 `Map` 信息至 QPID 队列，客户端 `MapReceiver` 获取 QPID 队列中消息，如图 8.4 所示，完成 `Map` 消息的传递。

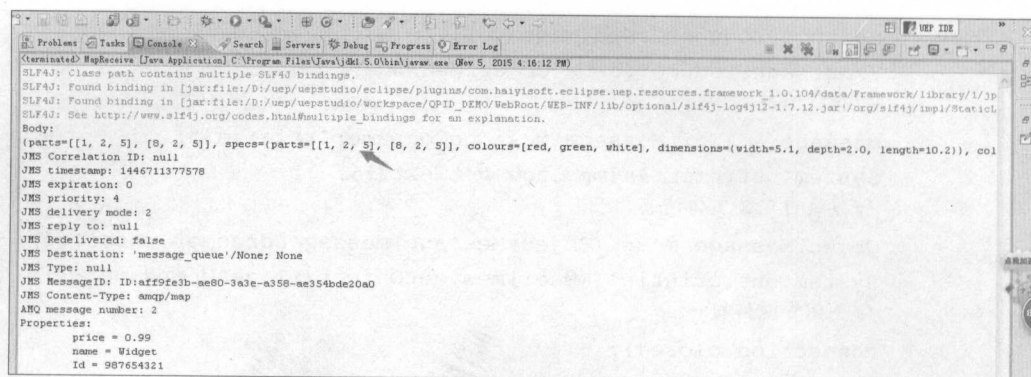


图 8.4 MapReceiver 获取 QPID 队列信息内容

客户端 MapSender.java (信息发送人) 示例代码如下:

```

package myf.caption8.qpid;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.jms.Connection;
import javax.jms.Destination;
import javax.jms.MapMessage;
import javax.jms.MessageProducer;
import javax.jms.Session;
import org.apache.qpid.client.AMQAnyDestination;
import org.apache.qpid.client.AMQConnection;
public class MapSender {
    public static void main(String[] args) throws Exception{
        //建立链接
        Connection connection =
            new AMQConnection("amqp://guest:guest@test/?brokerlist='tcp://
localhost:5672'");
        //获取 session
        Session session = connection.createSession(false, Session.AUTO_
ACKNOWLEDGE);
        //create: always, 表示如果队列不存在则创建
        Destination queue = new AMQAnyDestination("ADDR:message_queue;
{create: always}");
        //创建一个生产者, 用来发送消息
        MessageProducer producer = session.createProducer(queue);
        //创建一个 map 消息, 还有 textMessage 等
        MapMessage m = session.createMapMessage();
        //消息中放入各种键-值信息
        m.setIntProperty("Id", 987654321);
        m.setStringProperty("name", "Widget");
    
```

```

m.setDoubleProperty("price", 0.99);
//第一个对象
List<String> colors = new ArrayList<String>();
colors.add("red2222");
colors.add("green");
colors.add("white");
m.setObject("colours", colors);
//第二个对象
Map<String,Double> dimensions = new HashMap<String,Double>();
dimensions.put("length",10.2);
dimensions.put("width",5.1);
dimensions.put("depth",2.0);
m.setObject("dimensions",dimensions);
//第三个对象
List<List<Integer>> parts = new ArrayList<List<Integer>>();
parts.add(Arrays.asList(new Integer[] {1,2,5}));
parts.add(Arrays.asList(new Integer[] {8,2,5}));
m.setObject("parts", parts);
//第四个对象
Map<String,Object> specs = new HashMap<String,Object>();
specs.put("colours", colors);
specs.put("dimensions", dimensions);
specs.put("parts", parts);
m.setObject("specs", specs);

//      User user1 = new User("muyunfei","123456");
//      List<User> userList= new ArrayList<User>();
//      userList.add(user1);
//      m.setObject("user", userList);
//发送消息
producer.send(m);
//消息 ID
System.out.println(m.getJMSMessageID());
//关闭链接
connection.close();
}
}

```

客户端 MapReceiver.java（信息接收人）示例代码如下：

```

package myf.caption8.qpid;
import javax.jms.Connection;
import javax.jms.Destination;
import javax.jms.MapMessage;
import javax.jms.MessageConsumer;
import javax.jms.Session;
import org.apache.qpid.client.AMQAnyDestination;

```

```
import org.apache.qpid.client.AMQConnection;
public class MapReceiver {
    public static void main(String[] args) throws Exception
    {
        //创建连接
        Connection connection =
            new AMQConnection("amqp://guest:guest@test/?brokerlist='tcp://localhost:5672'");
        connection.start();
        //获取 session
        Session session = connection.createSession(false, Session.AUTO_
ACKNOWLEDGE);
        //获取队列
        Destination queue = new AMQAnyDestination("ADDR:message_queue;
{create: always}");
        MessageConsumer consumer = session.createConsumer(queue);
        //获取队列消息
        MapMessage m = (MapMessage)consumer.receive();
        //输出消息
        System.out.println(m.getStringProperty("name"));
        //输出消息
        System.out.println(m);
        //关闭连接
        connection.close();
    }
}
```

8.5.3 8080 端口问题

如果同时启动 JBoss 和 QPID，将出现端口冲突问题。原因是 JBoss 的 HTTP 服务端口号为 8080，与 QPID 的 HTTP 服务端口号相同。解决 JBOSS 与 QPID 的端口冲突方法和 JBoss 与 TOMCATE 端口冲突一样，具体如下：

- 01 找到 JBoss 文件 E:\jboss-4.2.3.GA\server\default\deploy\jboss-web.deployer\server.xml。
- 02 将其修改成未被占用的端口，如图 8.5 所示，修改 JBoss 端口。



备注：除修改 JBoss 端口外，也可以修改 QPID 端口：

```
$ ./qpid-server -prop "qpid.amqp_port=10000" -prop "qpid.http_port=10001"
```

查看端口是否被其他程序占用，可以在 cmd 命令窗口输入：

```
netstat -ano|findstr "8080"
```

```

<!-- Define a non-SSL HTTP/1.1 Connector on port 8080 -->
<Connector port="8089" address="0.0.0.0" URIEncoding="GBK"
maxThreads="250" maxHttpHeaderSize="8192"
emptySessionPath="true" protocol="HTTP/1.1"
enableLookups="false" redirectPort="8443" acceptCount="100"
connectionTimeout="20000" disableUploadTimeout="true" />

<!-- Define a SSL HTTP/1.1 Connector on port 8443
This connector uses the JSSE configuration, when using APR, the
connector should be using the OpenSSL style configuration
described in the APR documentation -->
<!--
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
maxThreads="150" scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS" />
-->

<!-- Define an AJP/1.3 Connector on port 8009 -->
<Connector port="8199" address="0.0.0.0" protocol="AJP/1.3"
emptySessionPath="true" enableLookups="false" redirectPort="8443" />

<Engine name="jboss.web" defaultHost="localhost">

<!-- The JAAS based authentication and authorization realm implementation
that is compatible with the jboss 3.2.x realm implementation.
- certificatePrincipal : the class name of the

```

图 8.5 MapReceiver 获取 QPID 队列信息内容

8.6 代理服务器

代理服务器是外网服务和内网服务之间的服务，外网服务通过代理服务器访问内网服务，类似于 WebService。与 WebService 的不同之处在于，它是使用数据流（输入流 `InputStream`、`OutputStream` 输出流）进行传递。使用数据流的优点是可以自定义数据流内容，可以是 JSON 格式的数据流，也可以是 XML 格式的数据流，还可以是二进制的协议内容。

以 JSON 格式数据流为例，代理服务器使用 `inputStream` 输入流获取数据信息，示例代码如下：

```

protected void doPost(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {
    //从输入流读取内容
    BufferedReader br = new BufferedReader(new InputStreamReader
(req.getInputStream()));
    String line = null;
    StringBuilder sb = new StringBuilder();
    while((line = br.readLine())!=null){
        sb.append(line);
    }
    //String content=URLDecoder.decode(sb.toString(), HTTP.ISO_8859_1);
    String content=sb.toString();
    //正则去除\
    content=content.replaceAll("\\\\", "\\\\");
    //获取调用方法
    String action =req.getParameter("action");
    String resultMsg="";//返回结果
    if("sendBusiMessage".equals(action)){
        //发送单个消息
    }
}

```



```

        resultMsg=sendBusiTextMessage(content);
    }else{
        resultMsg=createErrorMsg("invalid action method , illegal");
    }
    //返回输出结果
    OutputStream out = resp.getOutputStream();
    out.write(resultMsg.getBytes());
    out.flush();
    out.close();
    //resp.getOutputStream();
}

```



备注：可以使用二进制协议表示数据内容，如 001100001111，第一位是奇偶校验位，前四位表示设备，以此类推，来开发协议内容。

使用数据流发送请求内容，示例代码如下：

```

public boolean sendReq(String curbusiId, String userId, String busiFlowId) {
    boolean flag=false;
    //JSON 请求连接
    String jsonMessage="{\"busiId\": \""+curbusiId+"\", \"busiFlowId\": \""+busiFlowId+"\", \"userId\": \""+userId+"\"}";
    //从数据库获得请求连接
    String url = "https://....../..../";
    JSONObject responseBody=null;
    try {
        CloseableHttpClient httpclient = HttpClients.createDefault();
        HttpPost httpPost= new HttpPost(url);
        //发送 JSON 格式的数据
        StringEntity myEntity = new StringEntity(jsonMessage,ContentType.create("text/json", "GBK"));
        httpPost.setEntity(myEntity);
        ResponseHandler<JSONObject> responseHandler = new ResponseHandler<JSONObject>() {
            public JSONObject handleResponse(
                final HttpResponse response) throws ClientProtocolException,
                IOException {
                int status = response.getStatusLine().getStatusCode();
                if (status >= 200 && status < 300) {
                    HttpEntity entity = response.getEntity();
                    if(null!=entity){
                        String result= EntityUtils.toString(entity);
                        //根据字符串生成 JSON 对象
                        JSONObject resultObj = JSONObject.fromObject(result);
                        return resultObj;
                    }
                }
            }
        };
    }
}


```

```

        }else{
            return null;
        }
    } else {
        throw new ClientProtocolException("Unexpected status:
" + status);
    }
}

};
//返回的 JSON 对象
//httpPost.addHeader(HTTP.CONTENT_TYPE, "application/json");
responseBody = httpclient.execute(httpPost, responseHandler);
int result= (Integer) responseBody.get("errcode");
if(0==result){
    flag=true;
}else{
    flag=false;
}
}catch (Exception e) {
    System.out.println("获取数据失败");
    e.printStackTrace();
}
return flag;
}

```

 **备注：**本节演示的是文字的数据流传送，文件的数据流可以参考 8.8 节案例。对于文件的传送，除了流之外，还可以通过字符串的形式进行传送（通过文件输入流，获取 byte[]，将 byte 转成字符串的形式进行传送，获取后将 string 类型再次转换成 byte[]，注意 byte[] 不要直接 toString()）。

8.7 企业号服务 IP 白名单

对于环境要求严格的客户，则需要设置外网地址白名单。可以通过接口获取微信服务器 IP 端，来设置企业服务白名单。获取服务器 IP 端示例代码如下：

```

package com;
import java.io.IOException;
import net.sf.json.JSONArray;
import net.sf.json.JSONObject;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.ResponseHandler;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.CloseableHttpClient;

```

```

import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;
public class SafeMain {
    public static void main(String[] args) {
        //获取 Token
        String token=WxUtil.getToken("输入 corpId", "输入管理组凭证");
        System.out.println("token is "+token);
        try {
            CloseableHttpClient httpclient = HttpClients.createDefault();
            HttpPost httpPost= new HttpPost("https://qyapi.weixin.qq.com/cgi-bin/getcallbackip?access_token="+token);
            //创建响应处理
            ResponseHandler<JSONObject> responseHandler = new
ResponseHandler<JSONObject>() {
                public JSONObject handleResponse(
                    final HttpResponse response) throws
ClientProtocolException, IOException {
                    int status = response.getStatusLine().getStatusCode();
                    if (status >= 200 && status < 300) {
                        HttpEntity entity = response.getEntity();
                        if(null!=entity){
                            String result= EntityUtils.toString(entity);
                            //根据字符串生成 JSON 对象
                            JSONObject resultObj = JSONObject.fromObject
(result);

                            return resultObj;
                        }else{
                            return null;
                        }
                    } else {
                        throw new ClientProtocolException("Unexpected status: "
+ status);
                    }
                }
            };
            //返回的 JSON 对象
            JSONObject responseBody = httpclient.execute(httpPost,
responseHandler);
            //输出信息
            JSONArray arr=JSONArray.fromObject(responseBody.get("ip_list"));
            System.out.println("访问白名单如下所示:");
            for (int i = 0; i < arr.size(); i++) {
                System.out.println(arr.get(i));
            }
            httpclient.close();
        } catch (Exception e) {

```



```
// TODO Auto-generated catch block
e.printStackTrace();
}
}
```

输出微信 IP 端如图 8.6 所示。

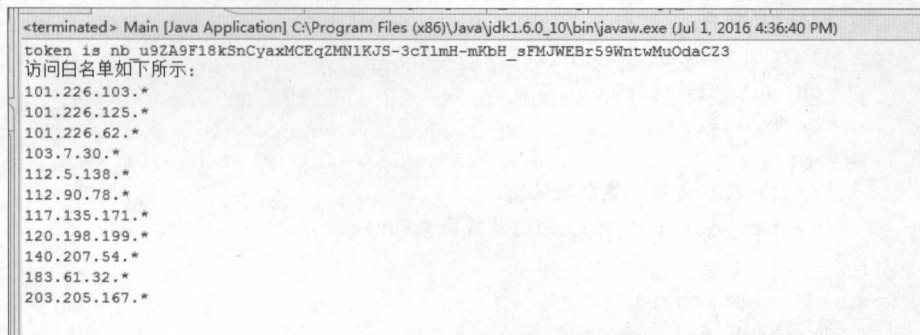



图 8.6 微信服务器 IP 端

 备注：允许访问外网 <https://qyapi.weixin.qq.com> 即可。

8.8 案例：通过 DMZ 服务器获取内网图片

图片是系统中最为常用的直观展示效果的元素，微信企业号如何获取内网数据中的图片呢？本节将以通过 DMZ 服务器获取内网图片为例，在读者学习获取内网图片的同时，更好地掌握内外网数据的安全提取。

(1) 微信客户端通过 DMZ 服务器获得内网图片资源。

微信通过 DMZ 区的外网服务器获得内网资源的输入流，使用输出流输出到当前页面（），示例代码如下：

```
//获取图片附件
public void getImageFile() throws UnsupportedEncodingException{
    HttpServletRequest req = ServletActionContext.getRequest();
    HttpServletResponse resp = ServletActionContext.getResponse();
    //设置编码格式
    req.setCharacterEncoding("UTF-8");
    resp.setCharacterEncoding("UTF-8");
    //从页面中获取的图片附件 ID
    String accessId = req.getParameter("accessId");
    //获得图片流
    try{
        //从 DMZ 服务器获取输入流
        InputStream in = getImageStream(accessId);
        if(null!=in){
            //获取输出流，输出到页面
```



```

        OutputStream outputStream = resp.getOutputStream();
        byte[] bytes = new byte[1024];
        int cnt=0;
        while ((cnt=in.read(bytes,0,bytes.length)) != -1) {
            outputStream.write(bytes, 0, cnt);
        }
        //关闭端口
        outputStream.flush();
        outputStream.close();
        in.close();
    }else{
        //图片获取失败, 显示默认图片
        System.out.println("图片获取失败");
    }
} catch (Exception e) {
    // TODO: handle exception
}
}
//从 DMZ 服务器获取输入流
public InputStream getImageStream(String accessFileId){
    boolean flag=false;
    // TODO Auto-generated method stub
    //JSON 请求链接
    //String jsonMessage="{\"accessFileId\":\""+accessFileId+"\"}";
    //获得链接
    String url ="http://内网 ip:8080/WXDEMO/imageAction.do?action"+
        "=getMySendImage&imgUrl=2015071717510761371.jpg";
    URL oaUrl;
    try {
        oaUrl = new URL(url);
        HttpURLConnection httpConn = (HttpURLConnection) oaUrl.openConnection();
        InputStream in = httpConn.getInputStream();
        return in;
    } catch (MalformedURLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return null;
}
}

```

(2) 内网服务从 FTP 中获取图片数据流。

内网服务主要从 FTP 中获取输入流, 并通过输出流输出至 DMZ 服务器中, 示例代码如下:

```
package com.sevlet;
```

```

import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.List;
import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import sun.net.TelnetInputStream;
import sun.net.ftp.FtpClient;
import com.WxUtil;
/**
 * 获取图片
 *
 * @author muyunfei
 *
 * <p>Modification History:</p>
 * <p>QQ      Author      Description</p>
 * <p>-----</p>
 * <p>1147417467      牟云飞      新建</p>
 */
public class ImageServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        this.doPost(request, response);
    }
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        // TODO Auto-generated method stub
        //获取调用方法
        String action = request.getParameter("action");
        String resultMsg = ""; //返回结果
        if("getImage".equals(action)){
            request.setCharacterEncoding("UTF-8");
            response.setCharacterEncoding("UTF-8");
            String accessId = request.getParameter("accessId");
            System.out.println("muyunfei:"+accessId);
            //数据库查询获取文件地址,以FTP为例
            String ftpUrl="aaaaa.jpg";
            //输出图片
            getFtpStream(ftpUrl, response);
        }
    }
}

```

```

    }else{
        resultMsg=createErrorMsg("invalid action method , illegal");
        //返回输出结果
        OutputStream out = resp.getOutputStream();
        System.out.println(resultMsg);
        out.write(resultMsg.getBytes());
        out.flush();
        out.close();
    }
}

/**
 * 生成错误信息
 * @param msg
 * @return
 */
private String createErrorMsg(String msg){
    String error="{\"errcode\": 1,\"errmsg\": \""+msg+"\"}";
    return error;
}

//获得 FTP 文件流
public void getFtpStream(String accessFileName,HttpServletResponse resp){
    //获取 FTP 连接字符串
    String ftpUrl="ftp://admin:666666@192.20.44.66:21\\gdfile";
    //ftp 链接字符串
    String ftpStr = "";
    if (ftpUrl.toUpperCase().indexOf("FTP://") > -1) {
        ftpStr = ftpUrl.substring(6, ftpUrl.indexOf("@"));
    } else {
        ftpStr = ftpUrl.substring(0, ftpUrl.indexOf("@"));
    }
    String ftpUserName = ftpStr.substring(0, ftpStr.lastIndexOf(":"));
    String ftpPWD = ftpStr.substring(ftpStr.lastIndexOf(":") + 1);
    ftpStr = ftpUrl.substring(ftpUrl.lastIndexOf("@") + 1);
    String ftpIP = ftpStr.substring(0, ftpStr.indexOf(":"));
    ftpStr = ftpStr.substring(ftpStr.lastIndexOf(":") + 1);
    String sep = File.separator;
    int ftpPort = Integer.parseInt(ftpStr.substring(0, ftpStr.indexOf(sep)));
    String filePath = ftpStr.substring(ftpStr.indexOf(sep) + 1);
    try {
        FtpClient ftpClient = new FtpClient(ftpIP, ftpPort);
        //userName、passWord 分别为 FTP 服务器的登录用户名和密码
        ftpClient.login(ftpUserName, ftpPWD);
        ftpClient.binary();
        ftpClient.cd(filePath);//path 为 FTP 服务器上保存上传文件的路径
        //输出流
        ServletOutputStream outputStream = resp.getOutputStream();
    }
}

```



```

//文件流
TelnetInputStream in = ftpClient.get(accessFileName);
byte[] bytes = new byte[1024];
int cnt=0;
while ((cnt=in.read(bytes,0,bytes.length)) != -1) {
    outputStream.write(bytes, 0, cnt);
}
outputStream.close();
in.close();
ftpClient.closeServer();
} catch (Exception e) {
    AppLogUtil.getAppLogger().error("上传文件出错!", e);
    e.printStackTrace();
}
}
}

```

(3) 通过字符串传送图片文件。

数据文件传送除通过数据流之外，还可以通过字符串的形式进行传送。实现步骤如下。

01 获取 FTP 图片资源数据流。

02 通过 byte[] 将数据流转换成 String 类型，示例代码如下：

```

BASE64Encoder encoder = new BASE64Encoder();
while ((b = inputstream.read(temp)) != -1) {
    fileStr+= encoder.encode(temp);
}

```

03 传递 String 字符串至 DMZ 服务器。

04 DMZ 服务器将 String 字符串解析成 byte[]，示例代码如下：

```

BASE64Decoder decoder = new BASE64Decoder();
byte[] appByte = decoder.decodeBuffer(fileStr);
ftpOutstream.write(appByte);

```

05 将 byte[]，通过输出流输出至前台展示。

字符串传递图片文件，示例代码如下：

```

package myf.caption8.fileString;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import net.sf.json.JSONObject;
import sun.misc.BASE64Decoder;
import sun.misc.BASE64Encoder;

```



```

/**
 * @author muyunfei
 *
 * <p>Modification History:</p>
 * <p>QQ      Author      Description</p>
 *
<p>-----</p>
 * <p>1147417467      牟云飞      新建</p>
 */
public class FileToString{
public static void main(String[] args) {
    FileToString demo = new FileToString();
    demo.sendPhotoByStr();
}
/**
 * 获得图片, 并将.jpg 通过 byte[] 保存成 string 进行传送
 * String 获得后转成 byte[], 然后保存成.jpg
 */
public String sendPhotoByStr() {
    try {
        //创建文件连接
        URL url = new URL("http://avatar.csdn.net/4/1/B/1_myfmyfmyfmyf.jpg ");
        HttpURLConnection httpURLConnection = (HttpURLConnection)
            url.openConnection();
        //设置连接参数
        httpURLConnection.setConnectTimeout(5 * 1000);
        httpURLConnection.setDoInput(true); //打开输入输出流
        httpURLConnection.setDoOutput(true);
        httpURLConnection.setUseCaches(false);
        httpURLConnection.setRequestMethod("GET");
        httpURLConnection.setRequestProperty("Charset", "UTF-8");
        httpURLConnection.setRequestProperty("Connection", "Keep-Alive");
        //建立连接
        httpURLConnection.connect();
        //获得输入流
        InputStream fis = httpURLConnection.getInputStream();
        //直接将流转换成字符串会出现异常, 只能先保存到本地, 然后进行转换
        String filename="photo";
        File file = new File(filename+".jpg");
        FileOutputStream outstream = new FileOutputStream(file);
        //将获得数据流转换成 byte, 然后转成 String, 向内网传送
        byte[] temp = new byte[1024];
        int b;
        while ((b = fis.read(temp)) != -1) {
            //          fileStr+=byte2hex(temp);
            //          fileStr+=new String(temp, "ISO-8859-1");

```

```

        outstream.write(temp, 0, b);
    }
    outstream.flush();
    outstream.close();
    fis.close();
    //获取本地文件转换成字符串
    File file2 = new File(filename+".jpg");
    FileInputStream inputStream=new FileInputStream(file2);
    //本地文件的输入流
    String fileStr="";//内网传送数据, file 的转 byte 的字符串
    BASE64Encoder encoder = new BASE64Encoder();
    while ((b = inputStream.read(temp)) != -1) {
        fileStr+= encoder.encode(temp);
    }
    inputStream.close();
    System.out.println("文件 1:"+fileStr);
    //删除本地文件
    file2.delete();
    //测试保存文件
    File ftpfile=new File("G:"+filename+"6666.jpg");
    FileOutputStream ftpOutstream = new FileOutputStream(ftpfile);
    BASE64Decoder decoder = new BASE64Decoder();
    byte[] appByte = decoder.decodeBuffer(fileStr);
    ftpOutstream.write(appByte);
    //    outstream.write(fileStr.getBytes("ISO-8859-1"));
    ftpOutstream.flush();
    ftpOutstream.close();
    System.out.println("ok");
    //成功向内网传送
    return "{\"code\": 1, \"result\": \""+fileStr+"\"}";
} catch (Exception e) {
    e.printStackTrace();
    return createErrorMsg("fail").toString();
}
}
/**
 * 生成错误信息
 * @param msg
 * @return
 */
private JSONObject createErrorMsg(String msg){
    String error="{\"code\": 1, \"result\": \""+msg+"\"}";
    return JSONObject.fromObject(error) ;
}
}

```

对结果集进行排序。

第 9 章

数据库及服务器

在系统开发中，数据库与服务器是必不可缺少的元素，同样，在微信企业号开发中，也不可少的需要数据库和服务器的支撑。数据库是组织、存储和管理数据的仓库，如 Oracle、MySQL、SQL Server 等，而服务器则是用于发布程序服务，提供运行支持的平台，如 Tomcat、JBoss、WebLogic 等。本章将从数据库基础知识以及部署过程中出现的问题来介绍如何正确使用数据库和服务器。本章知识点包括：

- SQL 语句基础：学习 SQL 语句、基础语句。
- HQL 基础语法：学习面向对象的数据库语句。
- 自定义函数处理：掌握 Hibernate 中自定义函数的识别处理。
- 数据库函数：以 Oracle 为例，介绍常用的数据库函数。
- 服务器：学习各种服务器的服务部署、堆栈溢出等问题。

9.1 常用 SQL 语句

在系统开发过程中，不论是何种数据库，都存在四种最基本的操作：查询、新增、更新和删除。本节将以这四种基本操作为例，介绍常用的 SQL 语句。

9.1.1 查询语句

select 用于查询数据库中的库表数据，查询结果存储于结果表中，语法如下：

```
SELECT 列名称 FROM 表名称
```

查询全部使用*号，示例语句如下：

```
SELECT * FROM 表名称
```

数据库 user 表数据见表 9.1 所示。

表 9.1 数据库user表数据


USER_ID	USER_NAME	MOBILE	EMAIL	WEIXIN_ID	DEPT
muyunfei	牟云飞	1556257xxxx	1147417467@qq.com	1147417467	1
zhangsan	张三	15599999999	zhangsan@163.com	zhangsan	6
zhangsi	张四	18666666666	zhangsi@163.com	zhangsi	6
lisi	李四	18577775555	lisi@126.com	lisi	8

(1) SQL 实例——查询 user 表中 MOBILE 字段为 1556257xxxx 的数据：

```
select * from user where mobile='1556257xxxx'
```

执行 SQL 语句，返回结果如下：

muyunfei	牟云飞	1556257xxxx	1147417467@qq.com	1147417467	1
----------	-----	-------------	-------------------	------------	---

 **备注：**字符串数值采用单引号的形式，虽然部门数据支持双引号，但还是建议读者使用单引号以保持兼容性。语句的执行并不区分大小写，建议书写时不要大小写混写。

(2) SQL 实例——查询 user 表中 MOBILE 字段包含 163 的数据：

```
select * from user where mobile like '%163%'
```

执行 SQL 语句返回结果如下：

zhangsan	张三	15599999999	zhangsan@163.com	zhangsan	6
zhangsi	张四	18666666666	zhangsi@163.com	zhangsi	6

数据库的模糊查询是使用%号进行查询，‘163%’表示以 163 开头的任意字符串，‘%163’则表示以 163 结尾的任一字符串。


(3) SQL 实例——查询 user 表中所有部门，不包含重复数据：

```
select distinct dept,count(a. dept) deptNum from user a group by a. dept
```

执行 SQL 语句返回结果如下：

DEPT	DEPTNUM
1	1
6	2
8	1

数据库中对于重复数据的合并可以使用关键词 distinct 进行处理，默认情况下是查询全部数据（关键词 all）。

 **备注：**其中变量 a、deptNum 为 user 库表的别名，可以使用 as 变量或者不写（默认追加）。group by 为分组语句，用于对数据进行分组处理，与之类似的是排序语句 order by，用于对结果集进行排序。

9.1.2 新增语句

insert 语句用于在库表中新增一条数据，语法如下：

```
INSERT INTO 表名称 VALUES (值1, 值2,....)
```

或者


```
INSERT INTO table_name (列1, 列2,...) VALUES (值1, 值2,....)
```

语法示例如下（向 user 库表中插入一条数据）：

```
INSERT INTO user VALUES ('wangwu', '王五', '15562571234', '15562571234@163.com', 'wangwu', 9)
```

执行结果如下：

USER_ID	USER_NAME	MOBILE	EMAIL	WEIXIN_ID	DEPT
muyunfei	牟云飞	1556257xxxx	1147417467@qq.com	1147417467	1
zhangsan	张三	15599999999	zhangsan@163.com	zhangsan	6
zhangsi	张四	18666666666	zhangsi@163.com	zhangsi	6
lisi	李四	18577775555	lisi@126.com	lisi	8
wangwu	王五	15562571234	15562571234@163.com	wangwu	9

 备注：新增、删除、修改语句需要对结果进行提交（commit），否则执行语句将被回滚（callback）。

9.1.3 更新语句

update 语句用于修改数据库中的数据，语法如下：


```
UPDATE 表名称 SET 列名称 = 新值, 列名称 = 新值 WHERE 列名称 = 某值
```

语法示例如下（修改 user 库表中 user_id 为 muyunfei 的手机号和邮箱）：

```
update user a set a.mobile='15566666666',a.email='15566666666@qq.com' where a.user_id='muyunfei'
```

执行结果如下：

USER_ID	USER_NAME	MOBILE	EMAIL	WEIXIN_ID	DEPT
muyunfei	牟云飞	15566666666	15566666666@qq.com	1147417467	1
zhangsan	张三	15599999999	zhangsan@163.com	zhangsan	6
zhangsi	张四	18666666666	zhangsi@163.com	zhangsi	6
lisi	李四	18577775555	lisi@126.com	lisi	8
wangwu	王五	15562571234	15562571234@163.com	wangwu	9

 **备注：**以 Oracle 为例，读者在修改、插入、查询数据时，遇到不能处理的特殊符号，如&、%等，可以通过 ASCII 值进行处理，示例如下（处理&符号，&对应的 ASCII 码为 38）：

```
update table set url='action.do?name=a||Chr(38)||'pwd=abc';---- action.do?name=a&pwd=abc
select 'Alibaba'||chr(38)||'Taobao' from dual;
```

如果不知道该特殊符号的 ASCII 值，则可以调用 ASCII 函数处理，示例如下：

```
select ascii('&') from dual;
```

9.1.4 删除语句

delete 语句用于删除库表中的某行数据，语法如下：

```
DELETE FROM 表名称 WHERE 列名称 = 值
```

语法示例如下（删除 user_name 为“牟云飞”的数据）：


```
delete from user a where a.user_name='牟云飞';
```

执行结果如下：

USER_ID	USER_NAME	MOBILE	EMAIL	WEIXIN_ID	DEPT
zhangsan	张三	15599999999	zhangsan@163.com	zhangsan	6
zhangsi	张四	18666666666	zhangsi@163.com	zhangsi	6
lisi	李四	18577775555	lisi@126.com	lisi	8
wangwu	王五	15562571234	15562571234@163.com	wangwu	9

9.2 HQL 语句基础语法

9.1 节我们学习了基本的 SQL 数据操作语句，本节将以 Hibernate 为例，以面向对象的方式书写 HQL 语句。

 **备注：**HQL，全称是 Hibernate Query Language，是 Hibernate 查询数据库的操作语言。

既然使用面向对象的方式，那么首先需要将数据库中表信息映射成相应的实体类，也称为 PO（Persisent Object，持久层对象），一般在编写 Hibernate 操作数据库时，都是以实体类、属性的方式进行编写，而 Hibernate 将识别解析面向对象的实体、属性，将其转化成对应的 SQL 来执行。

User 实体类如下：

```
public class User {
    private String userId;
    private String userName;
    private String mobile;
```

```
private String email;
private String weixinId;
private long dept;

public User(){}

public User(String userId, String userName, String mobile, String email,
            String weixinId, long dept) {
    super();
    this.userId = userId;
    this.userName = userName;
    this.mobile = mobile;
    this.email = email;
    this.weixinId = weixinId;
    this.dept = dept;
}

public String getUserId() {
    return userId;
}

public void setUserId(String userId) {
    this.userId = userId;
}

public String getUserName() {
    return userName;
}

public void setUserName(String userName) {
    this.userName = userName;
}

public String getMobile() {
    return mobile;
}

public void setMobile(String mobile) {
    this.mobile = mobile;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}


public String getWeixinId() {
    return weixinId;
}

public void setWeixinId(String weixinId) {
    this.weixinId = weixinId;
}
```

```

    }
    public long getDept() {
        return dept;
    }
    public void setDept(long dept) {
        this.dept = dept;
    }
}

```

 **备注：**Hibernate 对应的 XML 或注解读者可以查看 Hibernate 自行生成，如果想编写类似 Hibernate 面向对象的方式，则可以通过映射的方式实现，示例代码如下（参考链接 <http://blog.csdn.net/myfmyfmyfmyf/article/details/16805631>）：

```

Class cl = Class.forName(classPath);
//得到这个类的所有成员
Field[] name = cl.getDeclaredFields();
//得到这个类中所有的方法
Method[] method = cl.getDeclaredMethods();
//实例化
t = (T) cl.newInstance();
//调用 set 设置值
//method.invoke 的方式调用相应的函数

```


(1) HQL 中的 select 查询语句。

HQL 的查询语句与 SQL 查询语句基本相同，修改库表名为实体类，字段名为属性即可，如下所示：

```
select a from User a where mobile like '%163%'
```

或者

```
select new com.myf.User(a.userId, a.userName, a.mobile, a.email, a.weixinId,
a.dept) from User a where mobile like '%163%'
```

 **注意：**HQL 语句区分大小写，因此在书写实体类与对象时需要注意区分大小写。new com.myf.User 也可以写成 new User，不过还是建议写全文路径，以防止相同类名的问题。使用 new 产生对象时注意生成正确的构造函数，属性必须添加 get、set 方法。

(2) HQL 中 insert into 新增语句：

```
INSERT INTO User a VALUES ('wangwu', '王五', '15562571234', '15562571234@163.com',
'wangwu', 9)
```

(3) HQL 中 update 更新语句：

```
update User a set a.mobile='15566666666', a.email='15566666666@qq.com' where
```



```
a.userId='muyunfei'
```

(4) HQL 中 insert into 新增语句:

```
delete from User a where a.userName='牟云飞';
```

9.3 HQL 方言处理

HQL 语句中除了常见的函数 sum、count 之外, 必不可少的会出现自定义函数, 这时可以通过创建视图, 或者通过继承的方式实现。以 Oracle 为例, Hibernate 中 hibernate.dialect 默认处理类为 org.hibernate.dialect.Oracle10gDialect, 如下所示:

```
<prop key="hibernate.dialect">org.hibernate.dialect.Oracle10gDialect</prop>
```

为了使 Hibernate 能够通过自定义的函数而不检测, 我们需要继承 Oracle10gDialect 类, 创建 MyDialect 类, 如下所示:

```
package com.myf;
import org.hibernate.dialect.Oracle10gDialect;
import org.hibernate.dialect.function.SQLFunctionTemplate;
import org.hibernate.type.StringType;
/**
 * @author muyunfei
 * <p>Modification History:</p>
 * <p>QQ      Author      Description</p>
 * <p>-----</p>
 * <p>1147417467      牟云飞      新建</p>
 */
public class MyDialect extends Oracle10gDialect{
    public MyDialect(){
        super();
        //函数名必须是小写, 验大写时会出错
        //SQLFunctionTemplate 函数第一个参数是函数的输出类型, varchar2 对应 new
        StringType()    number 对应 new IntegerType()
        //?1 代表第一个参数, ?2 代表第二个参数。这时数据库 wx_f_get_partystyr 函数只需
        //要一个参数, 所以写成 wx_f_get_partystyr(?1)
        this.registerFunction("wx_f_get_partystyr", new SQLFunctionTemplate
(new StringType(), "wx_f_get_partystyr(?1)"));
        this.registerFunction("wx_f_get_party_codestr", new SQLFunctionTemplate
(new StringType(), "wx_f_get_party_codestr(?1)"));
        this.registerFunction("dbms_lob.substr", new SQLFunctionTemplate
(new StringType(), "dbms_lob.substr(?1)"));
        this.registerFunction("check_User_in_dept", new SQLFunctionTemplate
(new StringType(), "check_User_in_dept(?1,?2)"));
    }
}
```

在 MyDialect 构造函数中, 通过 registerFunction 注册自定义函数或其他 Hibernate 不识别的函数。

注意: 函数名必须是小写。SQLFunctionTemplate 函数中第一个参数是函数的输出类型, varchar2 对应 new StringType(), number 对应 new IntegerType(), ?1 代表第一个参数, ?2 代表第二个参数。

方言处理类完成之后, 修改 hibernate.dialect 原有配置, 将其修改为自定义的处理类, 示例代码如下:

```
<prop key="hibernate.dialect">com.haiyisoft.wx.web.struts.MyDialect</prop>
```

9.4 Tomcat 服务器

Tomcat 服务器, 是由 Apache 软件基金会开发的, 是一款免费的、开放源代码的 Web 应用服务器, 属于轻量级应用服务器, 是 Java 开发中常用的服务器。本节将从安装、部署及常见问题来介绍 Tomcat 在企业号开发中的应用。

9.4.1 在 SDK 中部署

下载 Tomcat 之后, 建议将 Tomcat 集成到开发工具中, 以方便开发。以 MyEclipse 为例, 单击【Window】|【Preferences】|【MyEclipse】|【Servers】|【Tomcat】, 或者在搜索框中输入“service”(如图 9.1 所示), 设置 Tomcat 所在目录, 设置完成之后, 设置 Tomcat 运行的 JDK, 如图 9.2 所示, 设置运行 JDK。

备注: 项目实施部署中不需要集成, 载入 war 包, 直接运行 bin 目录下的 startup.bat 即可。

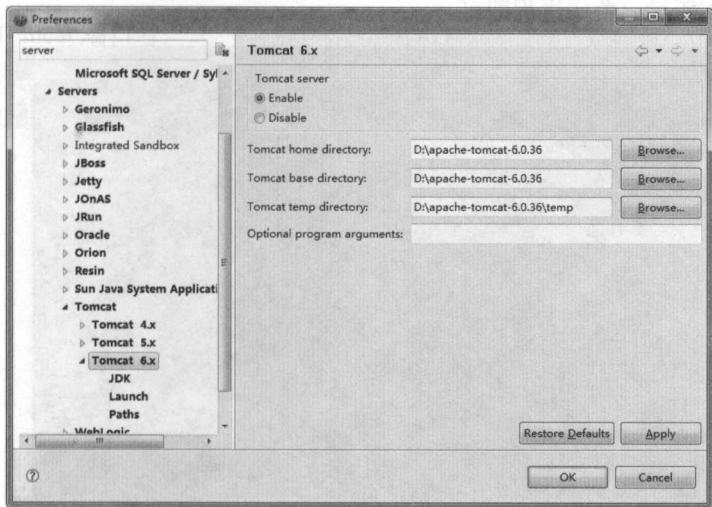


图 9.1 设置 Tomcat 路径

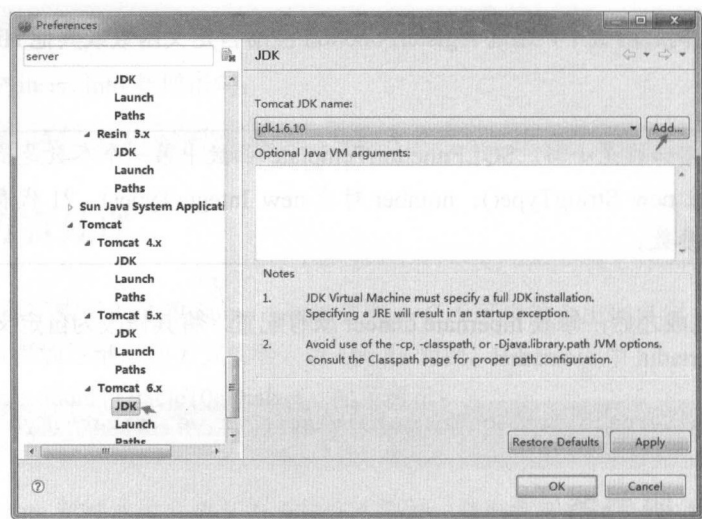


图 9.2 设置 Tomcat 运行 JDK

9.4.2 8080 端口号冲突

Tomcat 默认 HTTP 服务端口号为 8080，当出现服务冲突时，可以通过以下方式进行处理：

(1) 关闭其他 8080 端口服务。

通过 cmd 命令查看端口占用情况，输入：

```
netstat -ano|findstr "8080"
```

注意：端口号使用双引号。

查询出 8080 端口的占用程序，如果无占用程序，则不显示任何数据，如图 9.3 所示，获取占用端口的程序进程 PID，这里是 PID 为 10000 的进程。打开任务管理器，查找 PID 为 10000 的进程，如图 9.4 所示，关闭进程即可。

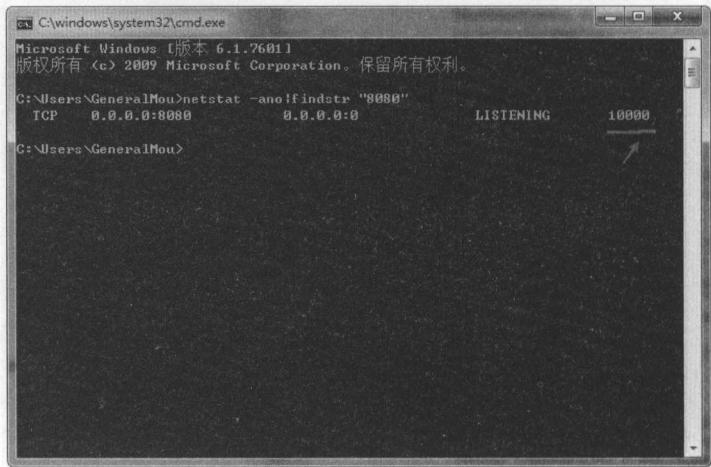


图 9.3 cmd 查询端口占用程序

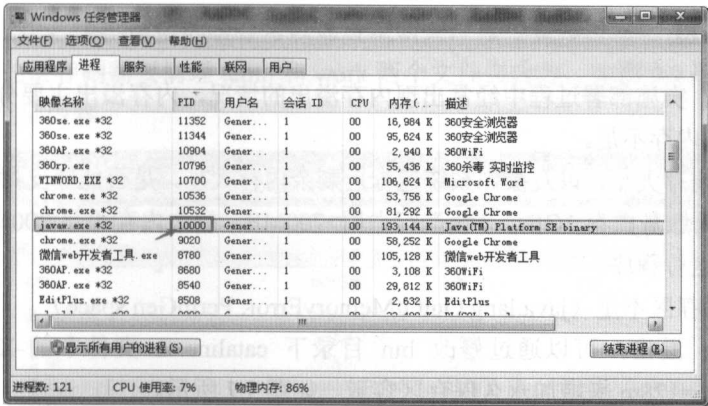


图 9.4 关闭端口占用进程

备注：javaw.exe 为 Java 虚拟机进程。

如果任务管理器不显示 PID，则可以单击【查看】|【选择列】，勾选 PID 即可，如图 9.5 所示。

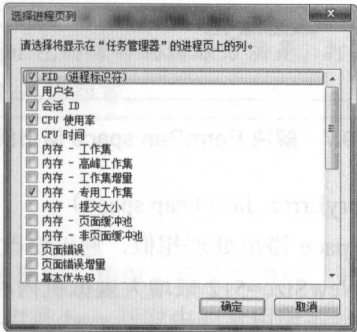


图 9.5 勾选 PID

(2) 修改 Tomcat 端口号

如果需要保留已有 8080 服务，则可以通过修改 Tomcat 端口号解决端口冲突，打开 apache-tomcat-7.0.52\conf 目录下的 server.xml 文件，修改 8080 端口，如图 9.6 所示。

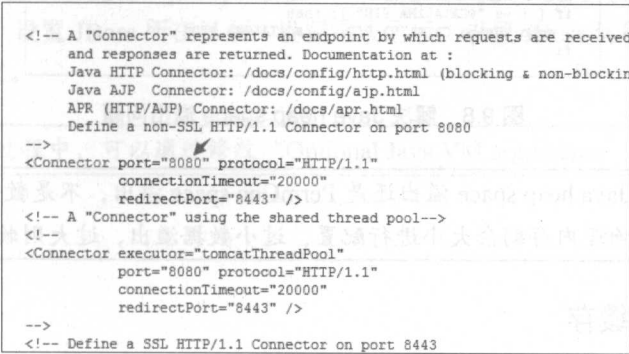


图 9.6 修改 Tomcat 服务端口号

9.4.3 内存调整

读者在开发、实施部署过程中经常出现内存溢出的情况，内存溢出主要分为三种情况。

(1) 物理机内存不足。

查看服务器内存大小，以及操作系统运行后剩余内存大小，是否能够支撑 Java 虚拟机的运行。例如，申请云服务内存 1GB，系统运行占用>700MB，剩余内存不足 300MB，因此 Java 虚拟机不足以正常运行程序。

(2) 永久保存区不足 (java.lang.OutOfMemoryError: PermGen space)。

PermGen space 溢出，可以通过修改 bin 目录下 catalina.sh 文件中的-XX:PermSize=64M -XX:MaxPermSize=128m 来增加永久保存区容量，如图 9.7 所示。

```
# Bugzilla 37848: only output this if we have a TTY
if [ $have_tty -eq 1 ]; then
  JAVA_OPTS="-server -Xms512m -Xmx512m -XX:MaxNewSize=512m -XX:PermSize=128M -XX:MaxPermSize=128m"
  echo "Using CATALINA_BASE:   $CATALINA_BASE"
  echo "Using CATALINA_HOME:   $CATALINA_HOME"
  echo "Using CATALINA_TMPDIR:  $CATALINA_TMPDIR"
  if [ "$1" = "debug" ]; then
    echo "Using JAVA_HOME:         $JAVA_HOME"
  else
    echo "Using JRE_HOME:          $JRE_HOME"
  fi
  echo "Using CLASSPATH:        $CLASSPATH"
  if [ ! -z "$CATALINA_PID" ]; then
    echo "Using CATALINA_PID:      $CATALINA_PID"
  fi
fi
```


图 9.7 解决 PermGen space 溢出问题

(3) Java 堆栈不足 (MemoryError: Java heap space)。

Java 堆栈溢出与 PermGen space 溢出处理相似，通过修改 bin 目录下 catalina.sh 文件中的-Xms512m -Xmx512m -XX:MaxNewSize=512 来增大虚拟机内存容量，如图 9.8 所示。

```
# Bugzilla 37848: only output this if we have a TTY
if [ $have_tty -eq 1 ]; then
  JAVA_OPTS="-server -Xms512m -Xmx512m -XX:MaxNewSize=512m"
  echo "Using CATALINA_BASE:   $CATALINA_BASE"
  echo "Using CATALINA_HOME:   $CATALINA_HOME"
  echo "Using CATALINA_TMPDIR:  $CATALINA_TMPDIR"
  if [ "$1" = "debug" ]; then
    echo "Using JAVA_HOME:         $JAVA_HOME"
  else
    echo "Using JRE_HOME:          $JRE_HOME"
  fi
  echo "Using CLASSPATH:        $CLASSPATH"
  if [ ! -z "$CATALINA_PID" ]; then
    echo "Using CATALINA_PID:      $CATALINA_PID"
  fi
fi
```

图 9.8 解决 Java heap space 溢出问题

 **备注：**不论是 Java heap space 溢出还是 PermGen space 溢出，不是数值越大越好，数值的大小需要根据物理内存剩余大小进行配置，过小数据溢出，过大则物理内存溢出。

9.4.4 清理数据缓存

数据缓存是程序开发中较为常见的问题，缓存将导致读者开发或修改的功能无法正常显示。

在微信企业号开发中，手机微信的缓存可以通过重新关注账号或等待微信次日清理缓存的方式清除，Tomcat 中的缓存则需要清理 temp 和 work 两个文件夹内容，如图 9.9 所示，分别打开两个文件夹，删除全部内容。

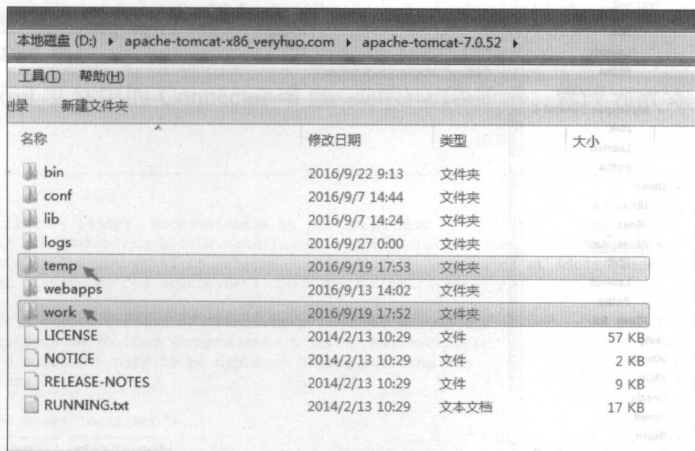



图 9.9 清理 Tomcat 缓存


 备注：class 文件的缓存或 jar 包的缓存的清除方法是，打开开发工具【Project】|【Clean】，选择当前工程，清除后重新编辑即可。

9.5 JBoss 服务器

JBoss 服务器与 Tomcat 服务器较为相似，都是免费的、开放源代码的 Web 应用服务器，HTTP 服务默认端口均为 8080，也是 Java 开发中的常用服务器。本节将从安装、部署以及常见问题来介绍 JBoss 在企业号开发中的应用。

9.5.1 JBoss 在 SDK 中安装

下载 JBoss 之后，为了更方便地开发，需要将 JBoss 集成到开发工具中。以 MyEclipse 为例，单击【Window】|【Preferences】|【MyEclipse】|【Servers】|【JBoss】，或者在搜索框中输入“server”（如图 9.10 所示），设置 JBoss 所在目录，设置完成之后，设置 JBoss 运行的 JDK，如图 9.11 所示，设置运行 JDK。

 备注：在开发过程中，可以通过修改“Optional Java VM arguments”来调整内存信息。

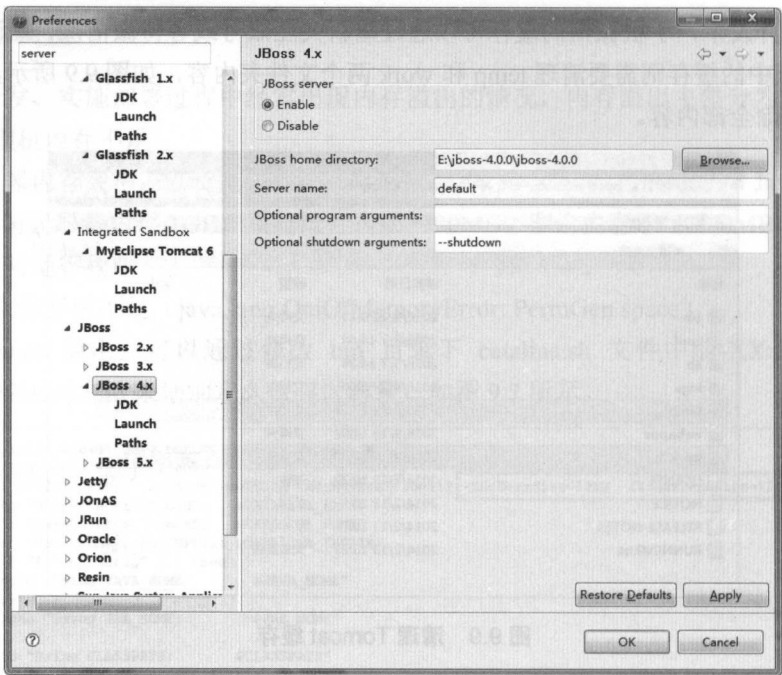


图 9.10 在 MyEclipse 中设置 JBoss

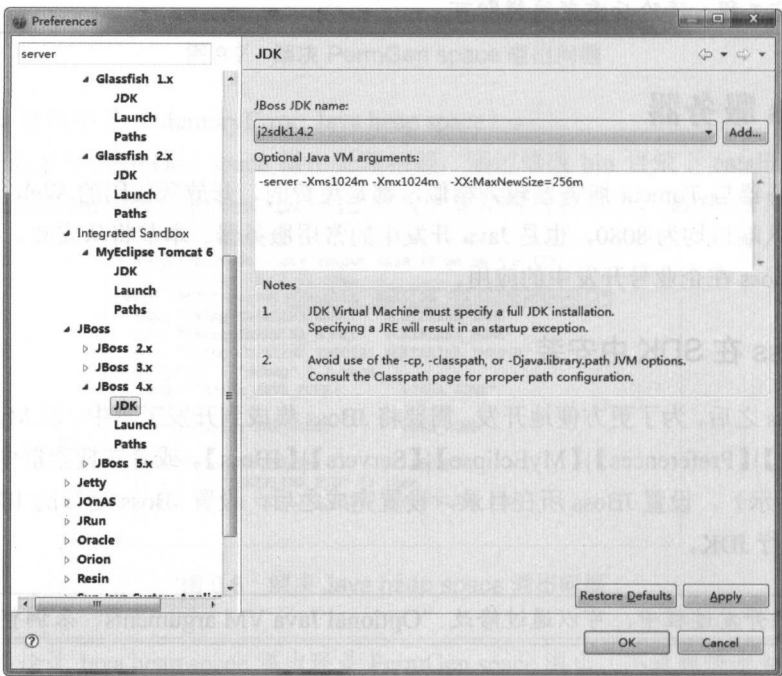


图 9.11 在 MyEclipse 中设置 JBoss JDK

9.5.2 修改 8080 端口

JBoss 默认 HTTP 服务端口号为 8080，与 Tomcat 提供的 HTTP 服务端口号相同。当出现服

务冲突时,可以通过 `netstat -ano|findstr "8080"` 命令查找相应进程并关闭。对于需要保留的其他 8080 端口服务,则可以通过修改 JBoss 端口号来解决端口占用问题,解决方法如下。

(1) 查找 `server.xml` 文件。

打开 JBoss 目录下 `server\default\deploy\jboss-web.deployer\` 目录,打开 `server.xml`。

(2) 修改 8080 端口。

查找 `server.xml` 文件中的 `Connector` 节点,如图 9.12 所示,修改 `port` 端口为其他非占用端口即可。

```
<Server>

<!--APR library loader. Documentation at /docs/apr.html -->
<Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on" />
<!--Initialize Jasper prior to webapps are loaded. Documentation at /docs/jasper-howto.html -->
<Listener className="org.apache.catalina.core.JasperListener" />

<!-- Use a custom version of StandardService that allows the
connectors to be started independent of the normal lifecycle
start to allow web apps to be deployed before starting the
connectors.
-->
<Service name="jboss.web">

    <!-- A "Connector" represents an endpoint by which requests are received
    and responses are returned. Documentation at :
    Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
    Java AJP Connector: /docs/config/ajp.html
    APR (HTTP/AJP) Connector: /docs/apr.html
    Define a non-SSL HTTP/1.1 Connector on port 8080
    -->
    <Connector port="8080" address="0.0.0.0" URIEncoding="utf-8"
        maxThreads="250" maxHttpHeaderSize="8192"
        emptySessionPath="true" protocol="HTTP/1.1"
        enableLookups="false" redirectPort="8443" acceptCount="100"
        connectionTimeout="20000" disableUploadTimeout="true" />

    <!-- Define a SSL HTTP/1.1 Connector on port 8443
    This connector uses the JSSE configuration, when using APR, the
    connector should be using the OpenSSL style configuration
    described in the APR documentation -->
    <!--
    <Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
        maxThreads="150" scheme="https" secure="true"
        clientAuth="false" sslProtocol="TLS" />
```

图 9.12 修改 JBoss 8080 端口号

9.5.3 JBoss 内存调整

JBoss 内存过大或过小引起的溢出,与 Tomcat 相同,也分为三种:物理内存不足、永久保存区内存不足 (`OutOfMemoryError: PermGen space`) 和 Java 堆栈不足 (`Java heap space`)。在物理内存运行的范围内,可以通过 “`-Xms256m -Xmx256m -XX:CompileThreshold=8000 -XX:PermSize=256m -XX:MaxPermSize=128m`” 解决 `PermGen space` 和 `Java heap space` 问题。解决示例如下。

01 修改 `bin` 目录下 `run.bat` 中的参数。

打开 `run.bat` 文件,修改 `JAVA_OPTS` 参数为 `set JAVA_OPTS=%JAVA_OPTS% -Xms256m -Xmx512m -XX:PermSize=256m -XX:MaxPermSize=128m`,如图 9.13 所示。

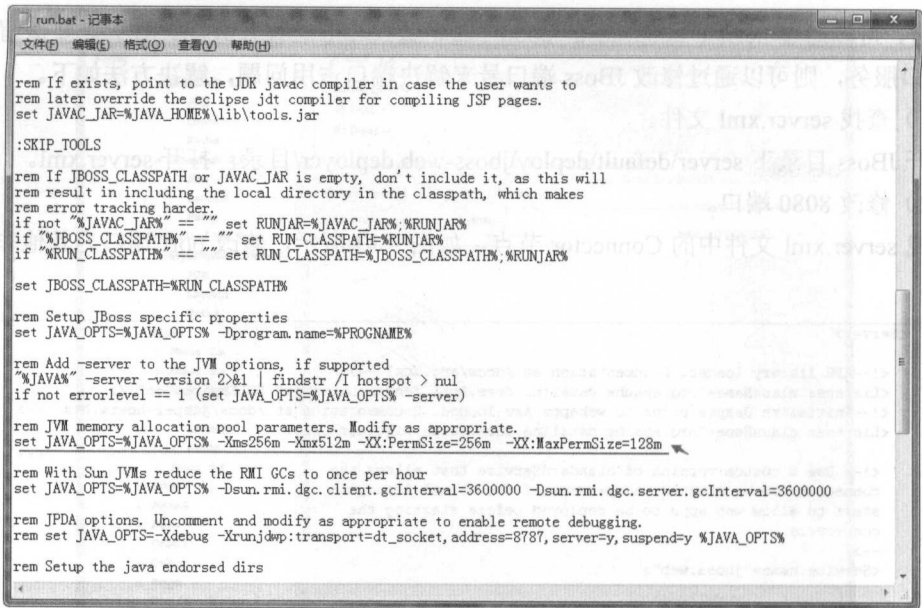


图 9.13 修改 run.bat 文件

02 修改 bin 目录下 run.config 中的参数。

打开 run.config 文件，修改 JAVA_OPTS 如图 9.14 所示。

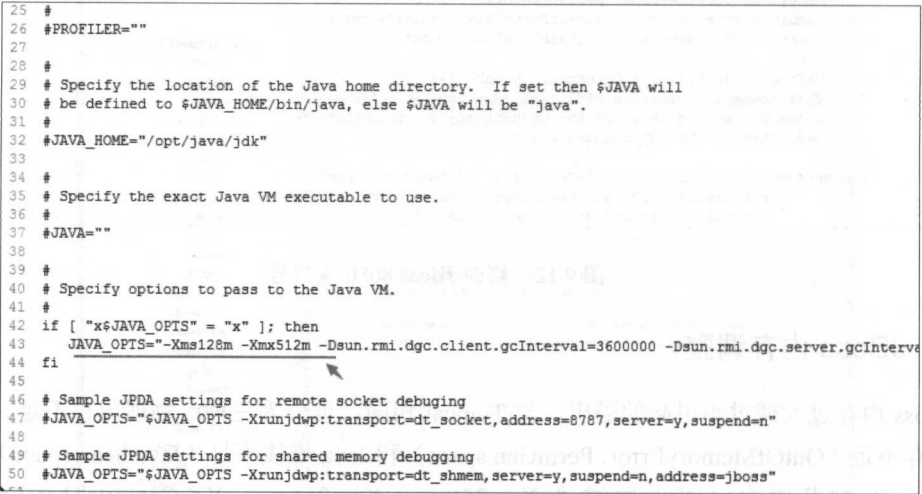


图 9.14 修改 run.config 文件

在分配内存容量时需要注意：

- ❑ Xms 表示 Java 内存堆最小值，Xmx 表示内存堆最大值，PermSize 表示永久区初始大小，MaxPermSize 表示永久区最大值。
- ❑ 最大值需要按需分配，不可以超过“物理机的实际内存大小”与“操作系统运行所占大小”的差值。
- ❑ run.bat 与 run.config 建议都配置，以减少操作系统而引起的异常。

9.5.4 发布缓存处理

JBoss 中的缓存处理与 Tomcat 缓存处理类似（在 9.4.4 节中已介绍过），需要清理 temp 和 work 两个文件夹内容。如图 9.15 所示，分别打开两个文件夹，删除全部内容。

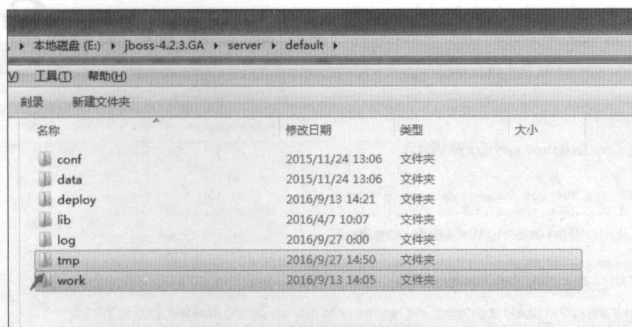


图 9.15 清理 JBoss 缓存

注意：清理发布缓存时，work 文件夹将缓存一部分页面信息，需要将 temp、work 同时清空。

9.6 WebLogic 服务器

WebLogic 服务器是 Oracle 公司出品的一个应用服务器，是一款收费的服务器，其默认 HTTP 服务端口为 7001。本节将为读者演示如何使用 WebLogic 发布服务以及常见问题的处理。

9.6.1 域的创建

WebLogic 服务器与 Tomcat、JBoss 服务器不同，在使用、部署之前必须创建一个特殊的空间——域（domain），项目的部署、发布、访问都需要依托于域进行设置。本节将为读者演示 WebLogic 服务器域空间的创建。

01 首先，单击【开始】|【WebLogic】，打开【Tools】|【Configuration Wizard】，如图 9.16 所示。

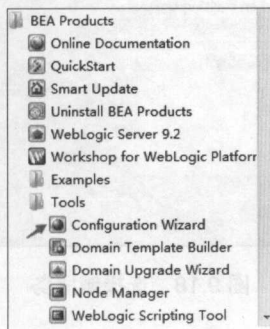


图 9.16 创建 domain

02 选择“创建一个新的 WebLogic 域（Create a new WebLogic domain）”，创建新的域空间，如图 9.17 所示。

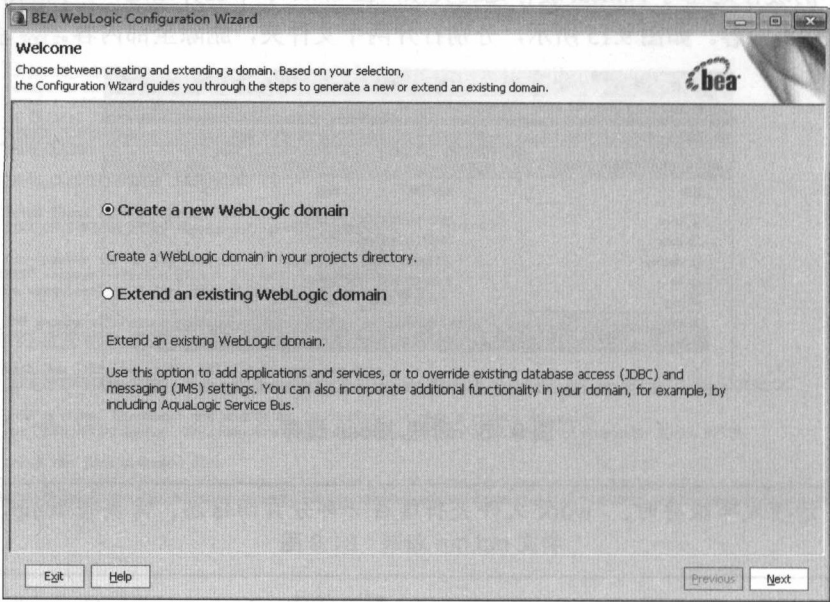


图 9.17 选择创建新的域空间

03 单击“下一步（Next）”按钮，选择域空间支持的 WebLogic 服务，默认是 WebLogic Servier (Required)，单击“Next”按钮，如图 9.18 所示。

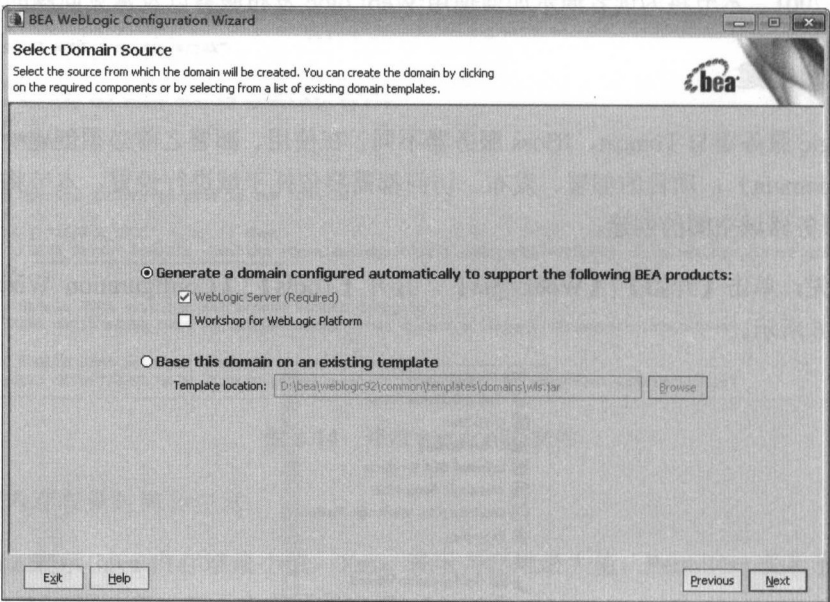


图 9.18 选择域服务

04 进入账户名、密码服务页面，填写当前 WebLogic 域服务的（启动、停止以及发布服

务等) 账户及密码, 如图 9.19 所示。



备注: 在 WebLogic12 版本中, 不支持以 Struts File 变量形式上传文件, 可以通过 Servlet 的形式或其他形式进行上传。

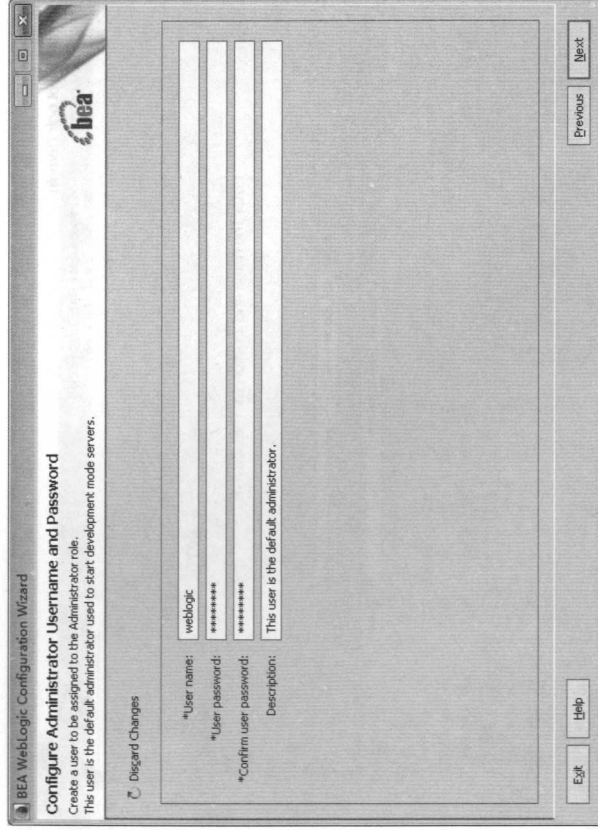


图 9.19 设置域空间账户和密码

05 选择 JDK, 默认选择 WebLogic 自带的 JDK, 建议使用 WebLogic 默认 JDK, 以减少不必要的异常, 如图 9.20 所示。

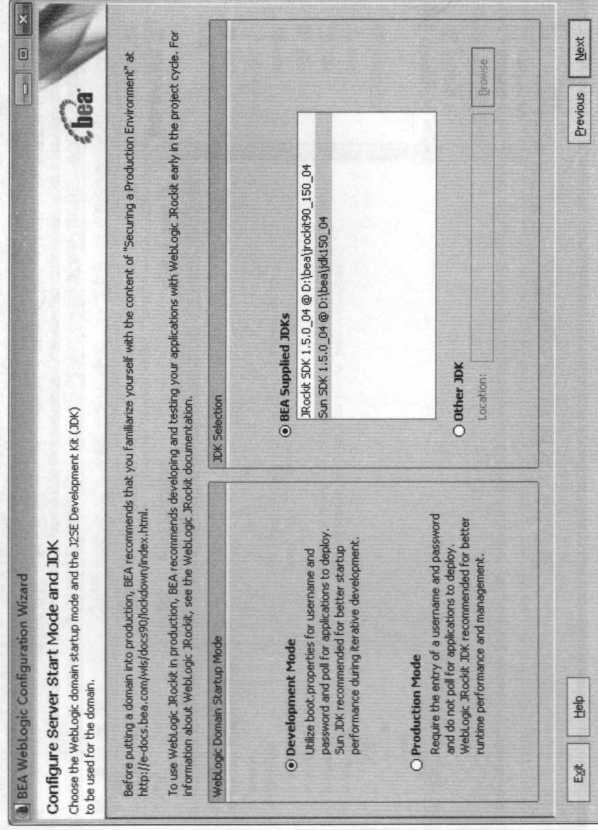



图 9.20 选择域空间虚拟机版本

06 选择完成 JDK 之后，将进入个性化选择页面，之后进入域名创建页面，创建自己的域空间名称即可，如图 9.21 所示。

 备注：域名的创建目录建议用非中文路径。

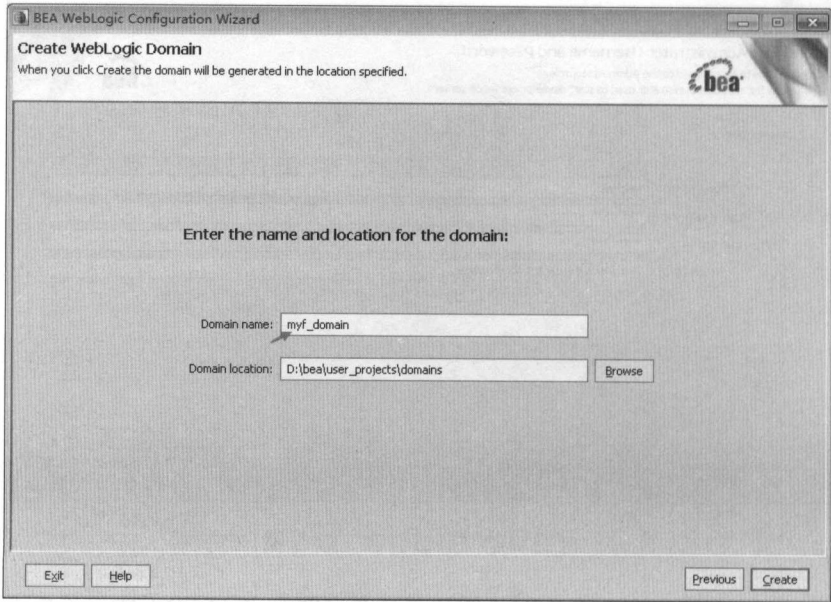


图 9.21 创建域名称

07 填写完名字之后，单击【create】按钮，进入创建界面，完成 WebLogic 域空间的创建，如图 9.22 所示。

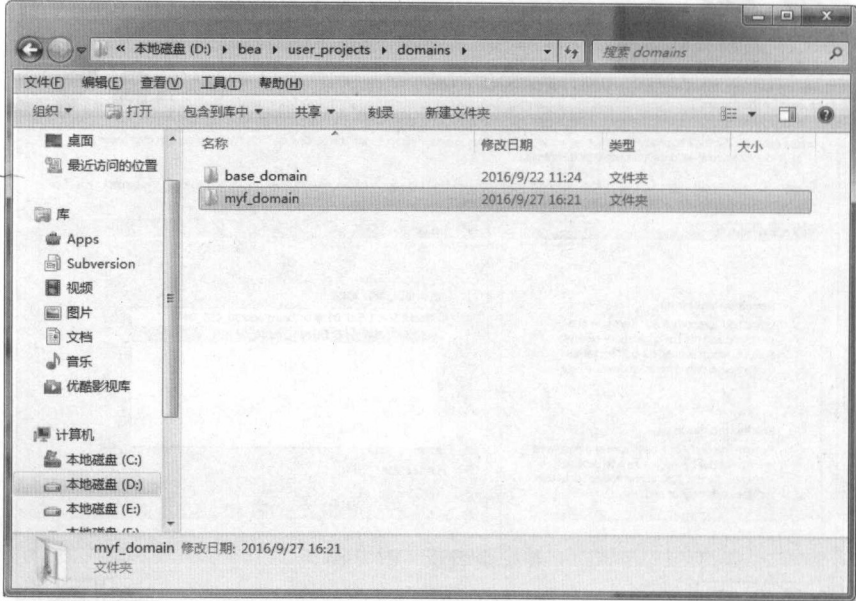


图 9.22 完成 myf_domain

9.6.2 WebLogic 在 SDK 中安装

读者下载 WebLogic 之后,需要将 WebLogic 中间件集成进开发工具中,方便读者开发调试,以 MyEclipse 为例,单击【Window】|【Preferences】|【MyEclipse】|【Servers】|【WebLogic】, (如图 9.23 所示) 或者在搜索框中输入“server”, 设置 WebLogic 所在目录, 设置完成之后, 设置 WebLogic 运行的 JDK, 如图 9.24 设置运行 JDK。

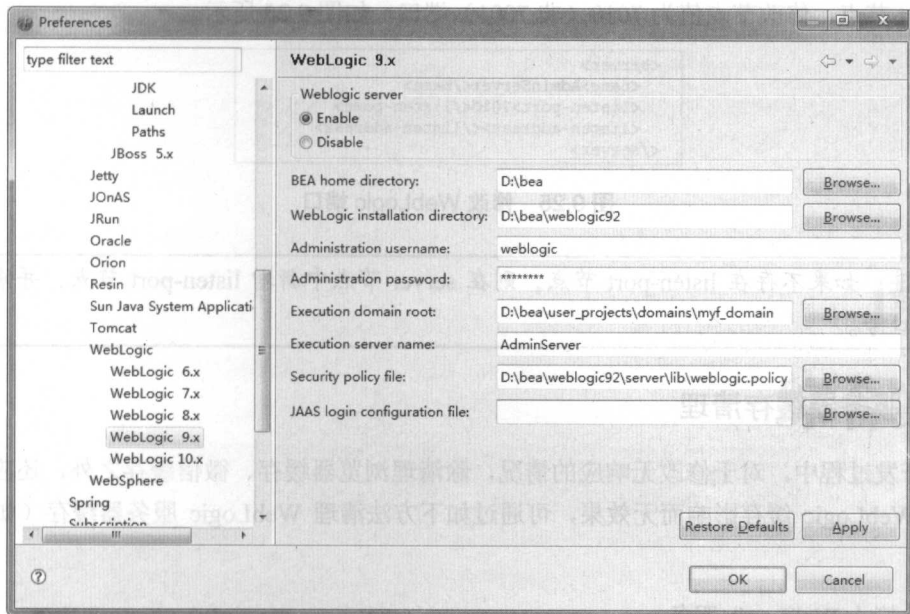


图 9.23 设置 WebLogic

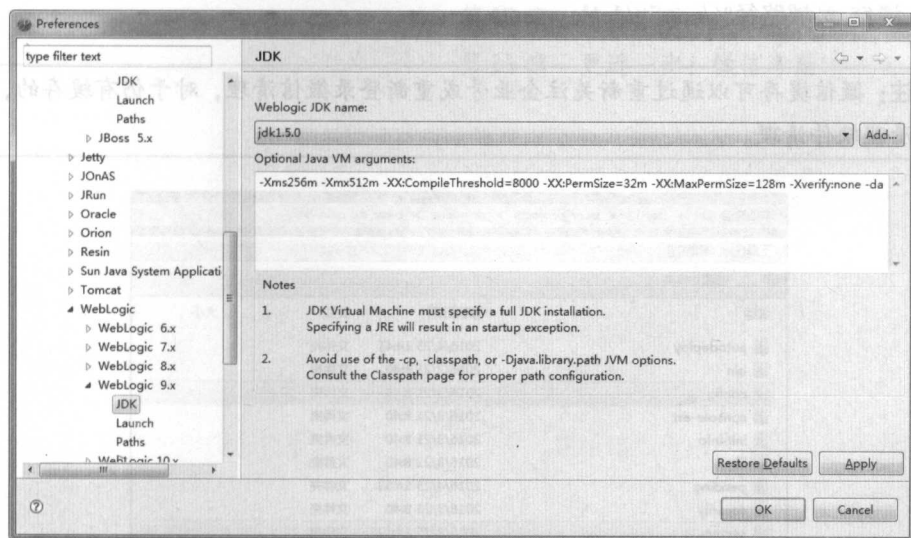


图 9.24 设置 WebLogic JDK


9.6.3 7001 端口号调整

WebLogic 默认端口号为 7001，当出现服务冲突时，可以通过 netstat -ano|findstr "7001" 命令查找相应进程，并关闭此进程。对于需要保留的其他 7001 端口服务，可以通过修改 WebLogic 端口号解决端口占用问题，解决方法如下：

打开\bea\user_projects\domains\%读者自定义的域名%\config\config.xml，查找 XML 中的 listen-port 节点，修改节点值为 7010（非 7001）端口，如图 9.25 所示。

```
<server>
  <name>AdminServer</name>
  <listen-port>7010</listen-port>
  <listen-address></listen-address>
</server>
```


图 9.25 修改 WebLogic 端口

 备注：如果不存在 listen-port 节点，则在 server 节点中新增 listen-port 节点，并修改相应端口号。

9.6.4 服务器缓存清理

在开发过程中，对于修改无响应的情况，除清理浏览器缓存、微信缓存之外，还应看看是否因为 WebLogic 缓存影响而无效果，可通过如下方法清理 WebLogic 服务器缓存（如图 9.26 所示）：

- 01 停止 WebLogic 服务。
- 02 清空%域路径%\servers\AdminServer/tmp 和 cache 文件夹。
- 03 清空 %域路径%\config\deployments。

 备注：微信缓存可以通过重新关注企业号或重新登录微信清理，对于仍有缓存的，可以使用清理软件清理。

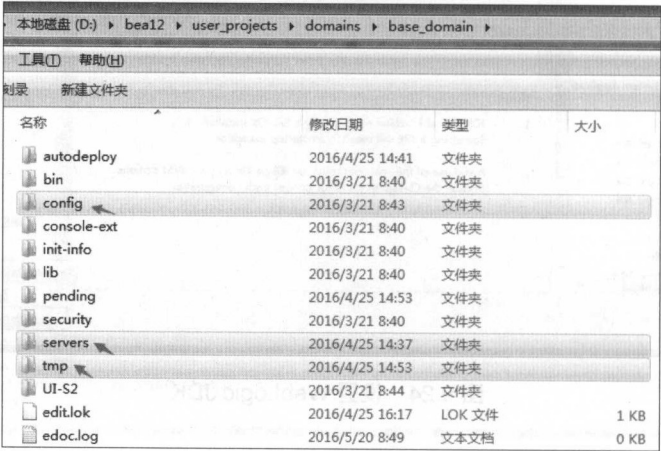


图 9.26 清理 WebLogic 缓存

第三篇

综合案例

第10章 基础应用——企业资讯

第11章 更进一步：微信考勤

第 10 章

基础应用——企业资讯

通过对前面几章的学习，相信你已经掌握了微信企业号开发的相关知识，本章将以企业日常工作为需求，开发基础的企业号应用——企业资讯。无论多大规模的企业，企业新闻、通知等信息都是员工经常接收的信息，接收的方式可能是纸质方式，也可能是即时通信的方式，还可能是口头的方式。由于微信的普及，现在员工通过微信也能够接收各种信息，包含公司新闻、通知公告、过程改进、研发动态以及公司制度等，如图 10.1 所示。



图 10.1 微信企业号基础应用——企业资讯

本章将系统地演示微信应用的创建，从而能够较快地创建自己的微信应用。企业资讯应用建设如下。

- 01 创建应用：在企业号微信管理端创建应用“企业资讯”，开启相应权限（参照 1.3 节）。
- 02 获取开发信息：获取应用 ID、管理组权限和企业号 ID（参照 1.2.7 节）。
- 03 开启回调：创建 Java 工程，开启回调模式（参照 4.2 节），并开启相关事件权限。
- 04 应用开发：创建员工首次进入信息推送，创建问答响应，推送菜单事件等操作。

10.1 创建应用

通过浏览器访问企业号微信管理端（<https://qy.weixin.qq.com>），并创建如图 10.2 所示的消息型应用——企业资讯，具体应用创建流程请参照 1.3 节。

备注：消息型应用功能较多，主页型应用主要采用 JSAPI 模式开发的页面应用，可以是 HTML 页面，也可以是 JSP、PHP 等页面，还可以是单页面应用（SPA）的主页面。



图 10.2 创建应用企业资讯

10.2 获取开发者信息

获取开发者信息主要分为应用基本信息和管理组权限信息，应用基本信息在开发中必需的信息是应用 ID，而在管理组权限信息中，则需要获得 CorpID（企业号唯一标识）及 Secret（管理组权限密钥），如图 10.3 所示。管理组创建流程可以参照 1.2.7 节。



图 10.3 获取企业资讯开发信息

注意：获取 Secret 时，当前管理组需要拥有当前应用的权限，如图 10.3 所示，“开发组”的管理员需要拥有应用（企业资讯）的管理权限。

10.3 开发实现

以 MyEclipse + MySQL 为例建立如图 10.4 所示的工程目录，工程名为 wx_demo，目录说明如下：

- dao 为数据访问层，实现 MySQL 数据库的操作。
- po 为持久层，用来保存数据库库表相关的实体类（本例以 Hibernate 生成）。
- QQTool 为微信提供的工具类，包括加密、解密等。
- service 为微信企业号服务请求的处理类。
- servlet 为微信企业号服务请求类，用于接收微信信息等。
- util 为工具类，包括封装微信接口服务等。
- vo 为虚拟的值对象，用于以对象形式传递数据。

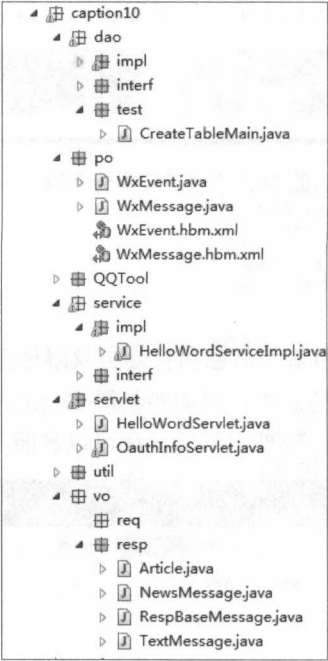


图 10.4 企业资讯开发实现

10.3.1 创建数据库 Table

创建数据库库表 WX_EVENT 和 WX_MESSAGE，用于记录微信操作的各种日志。在本案例中，主要用于实现对“首次”进入企业资讯的员工进行消息提醒，而非首次进入的员工则不提醒，SQL 语句如下：

```

/*
Navicat MySQL Data Transfer

Source Server          : MYSQL
Source Server Version  : 50021
Source Host            : localhost:3306
Source Database        : wxdb

Target Server Type     : MYSQL
Target Server Version  : 50021
File Encoding          : 65001

Date: 2016-10-08 21:59:55
*/

SET FOREIGN_KEY_CHECKS=0;
--
-- Table structure for `wx_event`
--
DROP TABLE IF EXISTS `wx_event`;
CREATE TABLE `wx_event` (
  `event_Id` bigint(20) NOT NULL auto_increment,
  `create_Time` datetime default NULL,
  `from_User` varchar(255) default NULL,
  `to_User` varchar(255) default NULL,
  `to_Party` varchar(255) default NULL,
  `to_Tag` varchar(255) default NULL,
  `agent_Id` int(11) default NULL,
  `event_Type` varchar(255) default NULL,
  PRIMARY KEY (`event_Id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Table structure for `wx_message`
--
DROP TABLE IF EXISTS `wx_message`;
CREATE TABLE `wx_message` (
  `msg_Id` bigint(20) NOT NULL auto_increment,
  `event_Id` bigint(20) default NULL,
  `title` varchar(255) default NULL,
  `content` varchar(255) default NULL,
  `description` varchar(255) default NULL,
  `media_Id` varchar(255) default NULL,
  `thumb_Media_Id` varchar(255) default NULL,
  `url` varchar(255) default NULL,
  `picurl` varchar(255) default NULL,

```



```
`safe` varchar(255) default NULL,
`author` varchar(255) default NULL,
`else1` varchar(255) default NULL,
`else2` varchar(255) default NULL,
PRIMARY KEY (`msg_Id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```



备注：创建数据库时需要注意与 Java 工程、服务器中间件编码一致，可以全部设置为 UTF-8，注意修改 MySQL 安装目录中的 my.ini 文件，将默认编码修改为 UTF-8。

10.3.2 生成 PO/VO 实体类

PO 实体类可以使用 Hibernate 直接生成。以 WX_EVENT 为例，在 po 中创建 WxEvent.java 类，对应持久层实体示例代码如下：

```
package myf.caption10.po;
/**
 * @author 牟云飞
 * @java Job 2012 年
 * @时间 2016-9-22
 */
import java.sql.Timestamp;

public class WxEvent {
    private static final long serialVersionUID = 1L;

    private Long eventId;
    private Timestamp createTime;
    private String fromUser;
    private String toUser;
    private String toParty;
    private String toTag;
    private Integer agentId;
    private String eventType;
    public Long getEventId() {
        return eventId;
    }
    public void setEventId(Long eventId) {
        this.eventId = eventId;
    }
    public Timestamp getCreateTime() {
        return createTime;
    }
    public void setCreateTime(Timestamp createTime) {
        this.createTime = createTime;
    }
}
```

```

public String getFromUser() {
    return fromUser;
}
public void setFromUser(String fromUser) {
    this.fromUser = fromUser;
}
public String getToUser() {
    return toUser;
}
public void setToUser(String toUser) {
    this.toUser = toUser;
}
public String getToParty() {
    return toParty;
}
public void setToParty(String toParty) {
    this.toParty = toParty;
}
public String getToTag() {
    return toTag;
}
public void setToTag(String toTag) {
    this.toTag = toTag;
}
public Integer getAgentId() {
    return agentId;
}
public void setAgentId(Integer agentId) {
    this.agentId = agentId;
}
public String getEventType() {
    return eventType;
}
public void setEventType(String eventType) {
    this.eventType = eventType;
}
}

```

Hibernate 实体映射关系，在 po 中创建 WxEvent.hbm.xml 文件，示例代码如下：

```


<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping package="myf.caption10.po">
    <class name="WxEvent" table="Wx_Event">
        <id name="eventId" column="event_Id" >

```

```

        <generator class="native"></generator>
    </id>
    <property name="createTime" column="create_Time" type="java.util.Date" >
</property>
    <property name="fromUser" column="from_User" type="java.lang.String" >
</property>
    <property name="toUser" column="to_User" type="java.lang.String" >
</property>
    <property name="toParty" column="to_Party" type="java.lang.String" >
</property>
    <property name="toTag" column="to_Tag" type="java.lang.String" >
</property>
    <property name="agentId" column="agent_Id" type="java.lang.Integer" >
</property>
    <property name="eventType" column="event_Type" type="java.lang.String" >
</property>
    </class>
</hibernate-mapping>

```

 **备注：**持久层实体类可通过 Hibernate 自动生成，建议读者手动写一个熟悉的 Hibernate 映射关系。手写**.hbm.xml 文件时，要注意在 hibernate.cfg.xml 文件中增加 mapping 节点，如<mapping resource="myf/caption10/po/WxEvent.hbm.xml" />。

本案例中的 VO 实体类，主要是响应消息的实体类，即以 RespBaseMessage.java 为父类生成的不同消息类（vo.resp 中创建）。RespBaseMessage 父类示例代码如下：

```

package myf.caption10.vo.resp;

/**
 * 响应消息，消息基类
 */
public class RespBaseMessage {
    private String ToUserName;
    private String FromUserName;
    private long CreateTime;
    // 消息类型
    private String MsgType;

    public String getToUserName() {
        return ToUserName;
    }

    public void setToUserName(String toUserName) {
        ToUserName = toUserName;
    }
}

```

```

public String getFromUserName() {
    return FromUserName;
}

public void setFromUserName(String fromUserName) {
    FromUserName = fromUserName;
}

public long getCreateTime() {
    return CreateTime;
}

public void setCreateTime(long createTime) {
    CreateTime = createTime;
}

public String getMsgType() {
    return MsgType;
}

public void setMsgType(String msgType) {
    MsgType = msgType;
}
}

```

文本消息响应类 `TextMessage.java` 示例代码如下：

```

package myf.caption10.vo.resp;

/**
 * 文本消息
 */
public class TextMessage extends RespBaseMessage {
    //回复的消息内容长度不可超过 2048 字节，否则微信服务器会放弃响应
    private String Content;

    public String getContent() {
        return Content;
    }

    public void setContent(String content) {
        Content = content;
    }
}

```

图文消息响应类 `NewsMessage.java` 和 `Article.java` 类示例代码如下：

```

package myf.caption10.vo.resp;

```



```
import java.util.List;

/**
 * news 消息
 *
 * @author muyunfei
 */
public class NewsMessage extends RespBaseMessage {
    //图文消息个数，限制在 10 条以内
    private int ArticleCount;
    //多条图文消息，默认第一个 item 为大图
    private List<Article> Articles;

    public int getArticleCount() {
        return ArticleCount;
    }

    public void setArticleCount(int articleCount) {
        ArticleCount = articleCount;
    }

    public List<Article> getArticles() {
        return Articles;
    }

    public void setArticles(List<Article> articles) {
        Articles = articles;
    }
}

package myf.caption10.vo.resp;

/**
 * 图文 model
 */
public class Article {
    //图文消息名称
    private String Title;
    //图文消息描述
    private String Description;
    //图片链接，支持 JPG、PNG 格式，较好的效果为大图 640*320（像素），小图 80*80（像素）
    private String PicUrl;
    //点击图文消息，跳转链接
    private String Url;
```

```

public String getTitle() {
    return Title;
}

public void setTitle(String title) {
    Title = title;
}

public String getDescription() {
    return null == Description ? "" : Description;
}

public void setDescription(String description) {
    Description = description;
}


public String getPicUrl() {
    return null == PicUrl ? "" : PicUrl;
}

public void setPicUrl(String picUrl) {
    PicUrl = picUrl;
}

public String getUrl() {
    return null == Url ? "" : Url;
}

public void setUrl(String url) {
    Url = url;
}
}

```

 **备注：**其他实体类（图片消息、语音消息、视频消息等）可通过继承 RespBaseMessage 类进行编写。

10.3.3 创建工具类 WxUtil

在 util 包中创建 WxUtil.java 类，用于控制配置信息和封装服务，示例代码如下：

```

package myf.caption10.util;

import java.io.Writer;
import java.util.Date;
import com.thoughtworks.xstream.XStream;
import com.thoughtworks.xstream.core.util.QuickWriter;

```

```

import com.thoughtworks.xstream.io.HierarchicalStreamWriter;
import com.thoughtworks.xstream.io.xml.PrettyPrintWriter;
import com.thoughtworks.xstream.io.xml.XppDriver;
import myf.caption10.vo.resp.Article;
import myf.caption10.vo.resp.NewsMessage;
import myf.caption10.vo.resp.TextMessage;

/**
 * @author 牟云飞
 * @tel 1556257xxxx
 * @QQ 1147417467
 * @java Job 2012 年
 * @时间 2016-9-22
 */
public class WxUtil {
    /**-----*/
    /*******主动发送消息不需要消息体进行加密, 格式为 JSON 格式******/
    /**-----*/
    //微信企业号唯一标识 CorpID
    public static final String MESSAGE_CORPID="wx0000000000000000";
    //主动调用: 管理组凭证密钥
    public static final String MESSAGE_SECRET=
"66666666666666666666666666666666";
    //主动调用: 发送消息获得 token
    public static String access_token;
    //主动调用: 请求 token 的时间
    public static Date access_token_date;
    //主动调用: 发送消息获得 token
    public static String jsapi_ticket;
    //主动调用: 请求 token 的时间
    public static Date jsapi_ticket_date;
    //微信会话: 发送消息获得 token
    public static String taking_access_token;
    //微信会话: 请求 token 的时间
    public static Date taking_access_token_date;
    //微信会话: 凭证密钥
    public static final String TAKING_MESSAGE_SECRET=
"99999999999999999999999999999999";
    /**-----*/
    /****被动响应消息需要对消息体先进行 AES 加密, 处理后再 base64 加密, 格式为 XML 格式***/
    /**-----*/
    //被动响应消息配置: token
    public static final String RESP_MESSAGE_TOKEN="muyunfeimyf";
    //被动响应消息配置: AES 密钥。 密钥生成规则: 32 位的明文经过 base64 加密后, 去掉末
    //尾=号, 形成 43 位的密钥
    public static final String RESP_MESSAGE_ENCODINGAESKEY=

```

```

"bXV5dW5mZWl3ZWl4aW5rYWlmYWppYW9jaGVuZ2l5Zm0";

/**-----*/
/*****消息类型*****/
/**-----*/
//消息类型：文本
public static final String MESSAGE_TYPE_TEXT = "text";
public static final String MESSAGE_TYPE_NEWS = "news";
/**-----*/
/*****消息事件类型*****/
/**-----*/
//消息类型：事件推送
public static final String MESSAGE_TYPE_EVENT = "event";
public static final String EVENT_TYPE_ENTERAGENT = "enter_agent";
/**-----*/
/*****微信应用 ID*****/
/**-----*/
public static final String AGENT_ID_ASSIST = "0";//默认应用“企业小助手”
public static final String AGENT_ID_HELLOWORD = "5";//企业资讯应用 ID
/**-----*/
/*****其他配置信息*****/
/**-----*/
//外网域名
public static String webUrl="http://myfmyfmyfmyf.vicp.cc/wx_demo";

/**
 * 扩展 xstream,使其支持 CDATA
 */
private static XStream xstream = new XStream(new XppDriver() {
    public HierarchicalStreamWriter createWriter(Writer out) {
        return new PrettyPrintWriter(out) {
            //对所有 XML 节点的转换都增加 CDATA 标记
            boolean cdata = true;

            @SuppressWarnings("unchecked")
            public void startNode(String name, Class clazz) {
                super.startNode(name, clazz);
            }

            protected void writeText(QuickWriter writer, String text) {
                if (cdata) {
                    writer.write("<![CDATA[");
                    writer.write(text);
                    writer.write("]]>");
                } else {
                    writer.write(text);
                }
            }
        };
    }
});

```



```

        }
    }
};


}

});

/**
 * 文本消息对象转换成 XML
 *
 * @param textMessage 文本消息对象
 * @return xml
 */
public static String messageToXml(TextMessage textMessage) {
    xstream.alias("xml", textMessage.getClass());
    return xstream.toXML(textMessage);
}

/**
 * 图文消息对象转换成 XML
 *
 * @param newsMessage 图文消息对象
 * @return xml
 */
public static String messageToXml(NewsMessage newsMessage) {
    xstream.alias("xml", newsMessage.getClass());
    xstream.alias("item", new Article().getClass());
    return xstream.toXML(newsMessage);
}
}

```

 **备注:** 部分配置信息可以放入数据库以方便读取, 可以放入 properties 文件中进行读取(参照 2.5 节)。

10.3.4 创建 Web 服务

创建 Web 服务/helloWordServlet.slt 服务 (http://域名/wx_demo/helloWordServlet.slt), 用于开启回调服务, 在 web.xml 中创建 servlet 服务, 示例代码如下:

```

<!-- 企业资讯应用的请求类, 回调模式使用-->
<servlet>
    <servlet-name>helloWordServlet</servlet-name>
    <servlet-class>
        myf.caption10.servlet.HelloWordServlet
    </servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>helloWordServlet</servlet-name>

```

```
<url-pattern>/helloWordServlet.slt</url-pattern>
</servlet-mapping>
```

HelloWordServlet 类, doGet 方法用于开启回调首次 GET 验证, doPost 用于接收微信信息, 示例代码如下:

```
package myf.caption10.servlet;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import myf.caption10.QQTool.WXBizMsgCrypt;
import myf.caption10.service.impl>HelloWordServiceImpl;
import myf.caption10.service.interf.IHelloWordService;
import myf.caption10.util.WxUtil;


/**
 * @author 牟云飞
 * @java Job 2012 年
 * @时间 2016-9-22
 */
public class HelloWordServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        //微信加密签名
        String signature = request.getParameter("msg_signature");
        //System.out.println("signature:"+signature);
        //时间戳
        String timestamp = request.getParameter("timestamp");
        //System.out.println("timestamp:"+timestamp);
        //随机数
        String nonce = request.getParameter("nonce");
        //System.out.println("nonce:"+nonce);
        //随机字符串
        String echostr = request.getParameter("echostr");
        //System.out.println("echostr:"+echostr);
        String sToken = WxUtil.RESP_MESSAGE_TOKEN;
        String sCorpID = WxUtil.MESSAGE_CORPID;
        String sEncodingAESKey = WxUtil.RESP_MESSAGE_ENCODINGAESKEY;
        try {
            WXBizMsgCrypt wxcpt = new WXBizMsgCrypt(sToken, sEncodingAESKey,
```

```
sCorpID);

        String sEchoStr; //需要返回的明文
        sEchoStr = wxcpt.VerifyURL(signature, timestamp,
                                    nonce, echostr);
        System.out.println("verifyurl echostr: " + sEchoStr);
        //验证 URL 成功, 将 sEchoStr 返回
        PrintWriter out = response.getWriter();
        out.write(sEchoStr);
        out.flush();
        out.close();
    } catch (Exception e) {
        //验证 URL 失败, 错误原因请查看异常
        e.printStackTrace();
    }
}

protected void doPost(HttpServletRequest request, HttpServletResponse
response)
    throws ServletException, IOException {
    //读取消息, 执行消息处理
    IHelloWordService service = (IHelloWordService) new HelloWordServiceImpl();
    service.receiveMsg(request, response);
}
}
```

 **备注:** doPost 用于中转消息, 消息的处理在 service 中进行, 实现业务的分离。

10.3.5 Service 处理 Web 请求

创建接口类 IHelloWordService, 定义业务处理方法, 示例代码如下:

```
package myf.caption10.service.interf;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public interface IHelloWordService {
    //回调模式处理消息
    public String receiveMsg(HttpServletRequest request, HttpServletResponse
response);
}
```

IHelloWordService 接口实现类 HelloWordServiceImpl.java, 用于实现以下功能:

- 向首次进入应用的员工发送文字消息, 如“欢迎使用微信企业资讯”等欢迎信息, 如图 10.5 所示。

- 响应员工发送的文本消息，能根据员工输入的信息进行查询并输出结果。
- 员工点击菜单按钮“最新资讯”，接收图文消息。

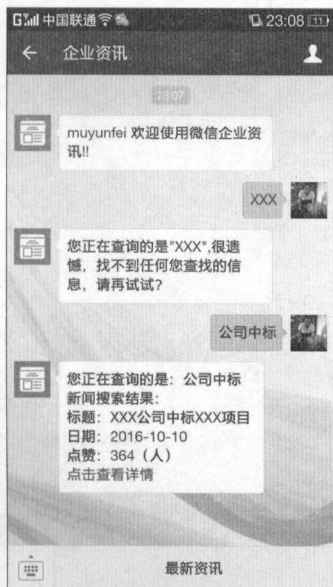


图 10.5 微信企业号基础应用

HelloWordServiceImpl 类示例代码如下：

```
package myf.caption10.service.impl;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.io.StringReader;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import javax.servlet.ServletInputStream;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import myf.caption10.QQTool.WXBizMsgCrypt;
import myf.caption10.dao.impl.WxEventDaoImpl;
import myf.caption10.dao.interf.IWxEventDao;
import myf.caption10.po.WxEvent;
import myf.caption10.service.interf.IHelloWordService;
import myf.caption10.util.WxUtil;
import myf.caption10.vo.resp.Article;
import myf.caption10.vo.resp.NewsMessage;
```



```

import myf.caption10.vo.resp.TextMessage;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;

/**
 * @author 牟云飞
 * @tel 1556257xxxx
 * @QQ 1147417467
 * @java Job 2012 年
 * @时间 2016-9-22
 */
public class HelloWordServiceImpl implements IHelloWordService {
    private IWxEventDao eventDao = new WxEventDaoImpl();

    //回调模式处理消息
    public String receiveMsg(HttpServletRequest request, HttpServletResponse
response) {
        //微信加密签名
        String sReqMsgSig = request.getParameter("msg_signature");
        //时间戳
        String sReqTimeStamp = request.getParameter("timestamp");
        //随机数
        String sReqNonce = request.getParameter("nonce");
        String sToken = WxUtil.RESP_MESSAGE_TOKEN;
        String sCorpID = WxUtil.MESSAGE_CORPID;
        String sEncodingAESKey = WxUtil.RESP_MESSAGE_ENCODINGAESKEY;
        try {
            //POST 请求的密文数据
            //sReqData = HttpUtils.PostData();
            ServletInputStream in = request.getInputStream();
            BufferedReader reader = new BufferedReader(new InputStreamReader(in));
            String sReqData = "";
            String itemStr = ""; //作为输出字符串的临时串, 用于判断是否读取完毕
            while (null != (itemStr = reader.readLine())) {
                sReqData += itemStr;
            }
            //对消息进行处理, 获得明文
            WXBizMsgCrypt wxcpt = new WXBizMsgCrypt(sToken, sEncodingAESKey,
sCorpID);
            String sMsg = wxcpt.DecryptMsg(sReqMsgSig, sReqTimeStamp, sReqNonce,
sReqData);
            //输出解密后的文件
            //For example:
            DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
            DocumentBuilder db = dbf.newDocumentBuilder();

```

```

StringReader sr = new StringReader(sMsg);
InputSource is = new InputSource(sr);
Document document = db.parse(is);
Element root = document.getDocumentElement();
//判断类型
NodeList nodelist_msgType = root.getElementsByTagName("MsgType");
String recieveMsgType = nodelist_msgType.item(0).getTextContent();
String content="";
String sEncryptMsg="";//加密的消息
if("text".equals(recieveMsgType)){
    //如果是文本消息……
    //获得内容
    NodeList nodelist1 = root.getElementsByTagName("Content");
    String resContent = nodelist1.item(0).getTextContent();
    //哪个员工进入
    NodeList enter_people_note = root.getElementsByTagName
("FromUserName");
    String enter_people = enter_people_note.item(0).getTextContent();
    //发送搜索结果消息
    //查询信息
    String resultTxtString="您正在查询的是\""+resContent+"\",很遗憾, "+
    "找不到任何您查找的信息,请再试试? ";
    //数据库查询信息
    String tempResultString = retrieveMsg(resContent);
    if(!"".equals(tempResultString)){
        resultTxtString = tempResultString;
    }
    TextMessage text = new TextMessage();
    text.setMsgType(WxUtil.MESSAGE_TYPE_TEXT);
    //发送者
    text.setFromUserName(WxUtil.MESSAGE_CORPID);
    //接收者
    text.setToUserName(enter_people);
    long time=new Date().getTime();
    text.setCreateTime(time);
    text.setContent(resultTxtString);
    String sRespData=WxUtil.messageToXml(text);
    //响应加密消息
    content=sEncryptMsg= wxcpt.EncryptMsg(sRespData, time+"", sReqNonce);
    //保存数据库
    //……
}else if("event".equals(recieveMsgType)){//如果是事件消息
    //获得事件类型
    NodeList nodelist1 = root.getElementsByTagName("Event");
    String event_type = nodelist1.item(0).getTextContent();
    if("enter_agent".equals(event_type)){//进入应用的事件
        //哪个员工进入

```

```

NodeList enter_people_note = root.getElementsByTagName
("FromUserName");

String enter_people = enter_people_note.item(0).getTextContent();
//判断是否第一次进入应用
List<WxEvent> enterList = eventDao.queryList(enter_people,
        Integer.valueOf(WxUtil.AGENT_ID_HELLOWORD)
        ,WxUtil.EVENT_TYPE_ENTERAGENT);
if(0==enterList.size()){
    //若是第一次进入,则推送一条消息
    //发送 HelloWorld 消息
    TextMessage text = new TextMessage();
    text.setMsgType(WxUtil.MESSAGE_TYPE_TEXT);
    //发送者
    text.setFromUserName(WxUtil.MESSAGE_CORPID);
    //接收者
    text.setToUserName(enter_people);
    long time=new Date().getTime();
    text.setCreateTime(time);
    text.setContent(enter_people+"欢迎使用微信企业资讯!!");
    String sRespData=WxUtil.messageToXml(text);
    //响应加密消息
    content=sEncryptMsg= wxcpt.EncryptMsg(sRespData,
time+"", sReqNonce);
}
//保存日志至数据库
try{
    WxEvent event = new WxEvent();
    event.setEventType("enter_agent");
    event.setFromUser(enter_people);
    event.setCreateTime(new Timestamp(new Date().getTime()));
    event.setAgentId(Integer.valueOf(WxUtil.AGENT_ID_
HELLOWORD));

    event.setToUser(WxUtil.MESSAGE_CORPID);
    eventDao.add(event);
}catch(Exception e){
    System.out.println("移动考勤进入应用,数据库保存失败");
    e.printStackTrace();
}
}
else if("click".equals(event_type)){
    //获得 event_key
    NodeList node_EventKey = root.getElementsByTagName("EventKey");
    String eventKey = node_EventKey.item(0).getTextContent();
    if("KEY_HELLO".equals(eventKey)){
        //发送 news 消息
        NewsMessage news=new NewsMessage();
        news.setFromUserName(WxUtil.MESSAGE_CORPID);//消息来源
    }
}

```



```

news.setMsgType(WxUtil.MESSAGE_TYPE_NEWS);
NodeList enter_people_note = root.getElementsByTagName
("FromUserName");

String enter_people = enter_people_note.item(0).getTextContent();
news.setToUserName(enter_people); // 回复人
long time = new Date().getTime();
news.setCreateTime(time); // 创建时间
news.setArticleCount(1); // 文章数量
List<Article> articles = new ArrayList<Article>();
Article article = new Article();
article.setTitle("最新资讯");
article.setPicUrl(WxUtil.webUrl+"/image/info.png");
article.setUrl(WxUtil.webUrl+"/help.html");
article.setDescription("XX 成功中标 XX 科技有限公司 XXX 项
目... ");

articles.add(article);
news.setArticles(articles);
String sRespData = WxUtil.messageToXml(news);
// 响应加密消息
content = sEncryptMsg = wxcpt.EncryptMsg(sRespData,
time+"", sReqNonce);
    }
}
}
//!!!!!!!!!!!!!!!!!!!!!!设置回复!!!!!!!!!!!!!!
//-----
if(null!=content&&!"".equals(content)){
    //输出
    PrintWriter out = response.getWriter();
    out.write(sEncryptMsg);
    out.flush();
    out.close();
}
} catch (Exception e) {
    //TODO
    //解密失败, 失败原因请查看异常
    e.printStackTrace();
}
return "";
}
//查询资讯信息
public String retrieveMsg(String searchText){
    String result = ""; //结果
    //读者根据自身需要编写响应查询
    //如果需要内外网转换, 请参照第 8 章
    if("公司中标".equals(searchText)){
        result="您正在查询的是: "+searchContext+" \r\n"

```



```
        +"新闻搜索结果: \r\n"
        +"标题: XXX 公司中标 XXX 项目\r\n"
        +"日期: 2016-10-10\r\n"
        +"点赞: 364 (人) \r\n"
        +"<a href='http://myfmyfmyfmyf.vicp.cc'>点击查看详情</a>";
    }
    return result;
}

public IWxEventDao getEventDao() {
    return eventDao;
}

public void setEventDao(IWxEventDao eventDao) {
    this.eventDao = eventDao;
}
}
```

 备注：回复消息为加密的 XML 消息，读者可以参照 4.3 节学习。

10.4 开启企业资讯应用回调

启动创建的 Java 工程，并将服务映射至外网（外网发布请参照 2.7 节），利用 Web 服务/helloWordServlet.slt 服务（http://域名/wx_demo/helloWordServlet.slt）以及 WxUtil 中的 RESP_MESSAGE_TOKEN 和 RESP_MESSAGE_ENCODINGAESKEY 开启回调模式（参照 4.2 节）。成功开启回调模式后，如图 10.6 所示，开启“上报进入应用事件”，用于员工每次进入时，微信企业号向员工服务推送 enter_agent 事件，通过数据库增加访问记录，开发首次进入的信息推送。

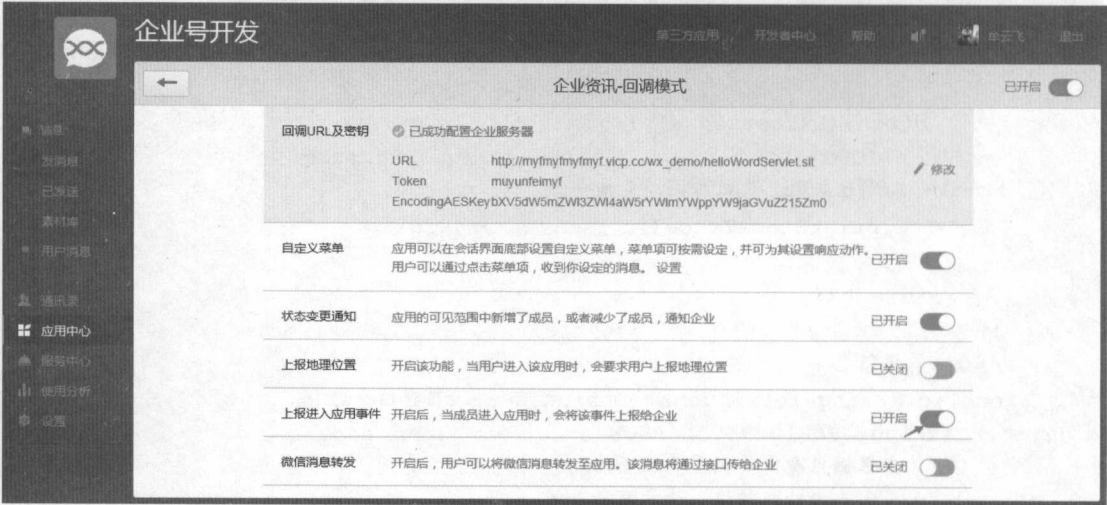



图 10.6 企业资讯回调模式

 备注：在 HelloWordServiceImpl 中处理 enter_agent 事件的具体方法，可参照 10.3.5 节。

10.5 创建最新资讯菜单

创建 KEY 值为 KEY_HELLO 的 click 事件菜单，点击菜单，通过回调链接，发送 KEY 值为 KEY_HELLO 的事件，如图 10.7 所示。菜单及其他类型菜单的创建可参照 4.4.5 节。

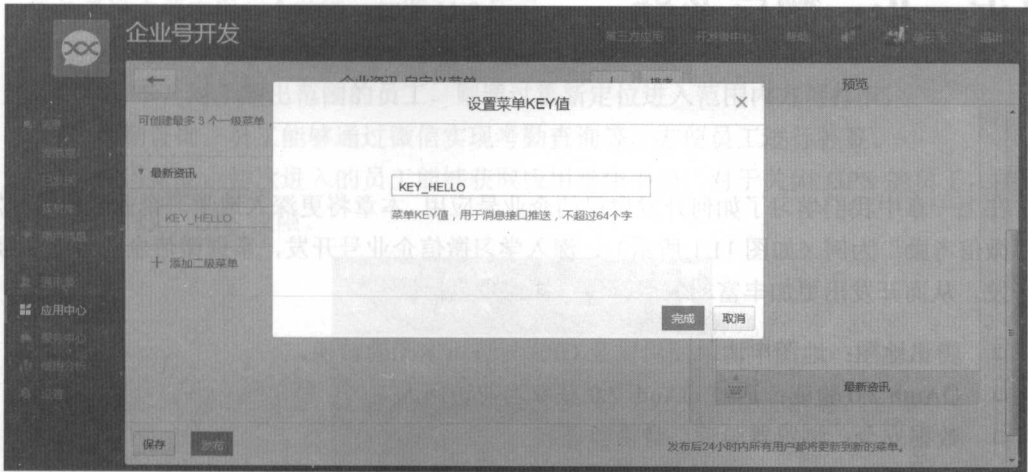



图 10.7 企业资讯应用创建菜单

 备注：在 HelloWordServiceImpl 中处理 click 事件，并通过 KEY 值发送图文（NewsMessage.java 和 Article.java）信息的具体方法，可以参照 10.3.5 节。菜单的新增、修改或者删除，都需要先进行发布，发布之后才可以使用。

10.6 本章小结

本章介绍了如何在企业号中创建应用、开发应用以及开启应用权限，并通过推送文字、图文消息的方式演示了微信企业号应用的开发。希望读者通过本章的案例能够掌握微信企业号开发的方法与技巧，最终能够创建自己的企业号应用。

第 11 章

更近一步：微信考勤

在上一章中我们学习了如何开发自己的企业号应用，本章将更深入地学习企业号应用开发，以“微信考勤”为例（如图 11.1 所示），深入学习微信企业号开发，掌握微信企业号的高级应用开发，从而开发出更加丰富的应用。本章知识点包括：

- 腾讯地图：地图申请、使用以及 GPS 与腾讯地图的转换。
- OAuth 2.0 验证：通过 OAuth 2.0 获取当前访问人。
- 数据安全：浏览器验证、身份验证。
- 回调模式：开启回调服务，响应菜单事件，进入应用等事件。
- JSAPI 模式：屏蔽微信右上角菜单，获取定位信息等。



图 11.1 微信考勤应用



备注：新创建的应用需要通过管理端增加相应管理组权限，否则将提示“redirect_uri unauthorized”，本章新创建的应用 ID 为 14，微信考勤应用的创建请参照 1.3 节。

11.1 场景回顾

企业考勤是 HR 管理中最为常见的功能，是员工每日上、下班经常操作的事情，企业需通过打卡机、App 等打卡方式监督员工每日按时工作。对于打卡机等物理设备来说，企业需要购买，增加了企业成本。移动设备存储空间，过多的 App 应用会影响移动设备的运行。微信轻应用，不仅解决了企业成本问题，而且解决了移动设备存储问题，方便员工操作。

微信考勤主要实现三个功能，如图 11.2 所示。

(1) 考勤打卡。初始化定位员工当前位置以及企业位置，在一定范围内，员工能够通过微信进行考勤打卡，对于超出范围的员工，则通过重新定位进入范围内方可打卡。

(2) 考勤查询。员工能够通过微信实现考勤查询等，方便员工进行补签。

(3) 使用帮助，初次进入的员工能够获取应用操作手册，对于关闭 GPS 的员工，能够方便地查看如何打开 GPS 权限。

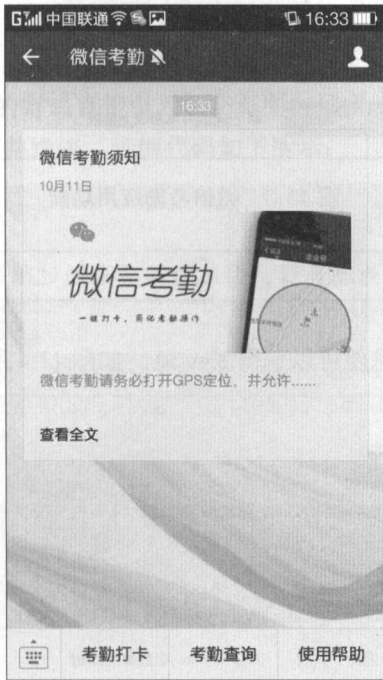


图 11.2 微信考勤应用功能



备注：读者可以根据自身需求开发其他功能，如考勤补签、休假申请等。

11.2 腾讯地图引入

微信考勤主要是基于地图实现考勤功能，为了更好地与微信兼容，我们采用腾讯地图进行开发。开发之前首先需要申请腾讯地图 KEY，并掌握地图的使用方式，本节介绍如何使用腾讯地图。

11.2.1 腾讯地图 Key 申请

腾讯地图 Key，用于地图 Key 鉴权，完全免费申请，免费使用，除调用街景接口必须使用 key 外，地图和服务接口都没有强制限制。腾讯地图在没有 Key 的情况下也能够使用，但建议申请 Key，以防止后期在无 key 的情况下影响使用。

读者可以通过 <http://lbs.qq.com/key.html> 申请地址，填写申请信息。填写信息如图 11.3 所示，选择应用类型为浏览器即可。

开发者密钥申请

* 应用名称:

微信考勤

描述:

微信企业号微信考勤

* 应用类型:

☒ 浏览器

（调用Javascript API、JSONP调用WebService API、静态图API等）

☐ 移动端


（调用Android/iOS SDK产品、WebService API或静态图API等）

☐ 服务端

（调用WebService API、静态图API等）

* 验证码:

wthz



看不清，换一张

提交

☒ 已阅读并同意以上条款

图 11.3 微信考勤应用功能

 备注：由于勾选应用类型为浏览器，因此微信考勤通过微信内置浏览器访问读者服务。

填写完成信息后，提交信息便可以获得 Key 值，如图 11.4 所示。

您的开发者key创建成功

QWZBZ-JHR3R-IN4W2-WCRIW-AOTYV-DUBE7

Key使用方法请参见：

[JavaScript API key设置使用指南]

[WebService API key设置使用指南]

[Android SDK key设置使用指南]

[iOS SDK key设置使用指南]

完成

key设置

图 11.4 腾讯地图 Key 值

如果已经拥有 Key 值，则可以在腾讯地图开放平台获取相应信息，如图 11.5 所示。

394



图 11.5 腾讯地图开发平台

11.2.2 腾讯地图 Demo

地图加载可通过在线地图的方式直接引入，在页面中预留 div 元素，并在浏览器的文档对象模型（DOM）中获取此元素并操作，示例代码如下所示：

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>腾讯地图示例</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0,
minimum-scale=1.0, maximum-scale=1.0, user-scalable=no"/>
<style type="text/css">
html,body{
width:100%;
height:100%;
}
*{
margin:0px;
padding:0px;
}
body, button, input, select, textarea {
font: 12px/16px Verdana, Helvetica, Arial, sans-serif;
}
p{
width:603px;
padding-top:3px;
overflow:hidden;
}
.btn{
width:142px;
}
```

```
#container{
    min-width:600px;
    min-height:767px;
}
</style>
<script charset="utf-8" src="http://map.qq.com/api/js?v=2.exp"></script>
<script>
window.onload = function(){
    //初始化地图函数，自定义函数名 init
    function init() {
        //定义 map 变量， 调用 qq.maps.Map() 构造函数，获取地图显示容器
        var map = new qq.maps.Map(document.getElementById("container"), {
            center: new qq.maps.LatLng(39.916527,116.397128),
            // 地图的中心地理坐标

            zoom:8    // 地图缩放大小
        });
    }
    //调用初始化函数地图
    init();
}
</script>
</head>
<body>
<!-- 定义地图显示容器 -->
<div id="container"></div>
</body>
</html>
```

执行效果如图 11.6 所示。

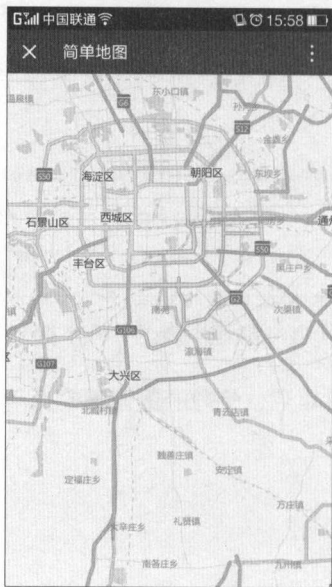




图 11.6 腾讯地图开发平台

 **备注：**腾讯地图也可以通过异步加载的方式进行加载本节使用的是直接加载，如需异步加载，则可通过官网查看（http://lbs.qq.com/javascript_v2/guide-start.html）。

11.2.3 腾讯地图坐标转换

腾讯地图提供了 `qq.maps.convertor` 对象用于坐标转换，其中 `translate` 方法 `translate(points: LatLng | Point | Array.<LatLng> | Array.<Point>, type: Number, callback: Function)`，用于将其他地图服务商的坐标批量转换成腾讯地图经纬度坐标。`type` 用于说明是哪个服务商的坐标。`type` 的可选值为：1. `gps` 经纬度；2. 搜狗经纬度；3. 百度经纬度；4. `mapbar` 经纬度；5. `Google` 经纬度；6. 搜狗墨卡托。示例代码如下：

```
<script charset="utf-8" src="http://map.qq.com/api/js?v=2.exp&libraries=
convertor"></script>
<script>
function init(){
    var map = new qq.maps.Map(document.getElementById("container"),{
        center: new qq.maps.LatLng(39.916527,116.397128),
        zoom: 13
    });
    //转换百度坐标为腾讯坐标
    qq.maps.convertor.translate(new qq.maps.LatLng(39.911082,116.396135),
3, function(res){
        latlng = res[0];
        var marker = new qq.maps.Marker({
            map : map,
            position : latlng
        });
    });
});
}:
```

 **备注：**微信企业号 JS-SDK 中提供的地理位置接口，默认获取的为 `GPS` 坐标。如使用其他地图，则需要进行相应的转换。

11.3 开发实现

开发实现上与企业资讯基本相同，目录如图 11.7 所示，分为存在，`index.jsp`、`registerMobile.jsp`，`jsp` 属于页面存放在 `WebRoot` 中、`action`、`servlet`、`vo` 以及 `service`，方便读者更好的阅读以及实现。

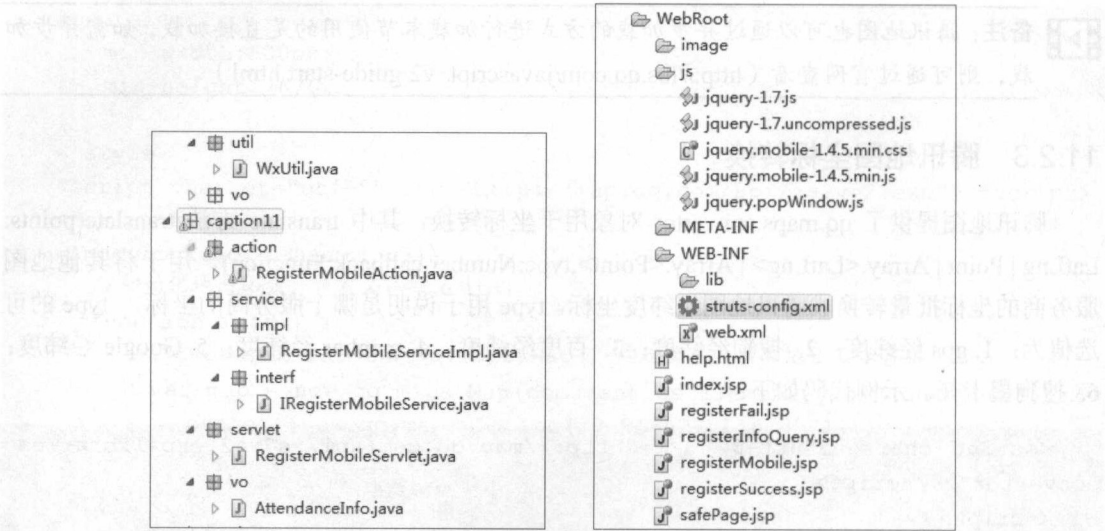


图 11.7 开发实现

11.3.1 创建微信工具类

在企业资讯工具类 WxUtil.java 的基础上增加微信考勤方法，示例代码如下：

```
package myf.caption10.util;

import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.io.Writer;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Date;
import java.util.Formatter;
import java.util.HashMap;
import java.util.Map;
import java.util.UUID;
import net.sf.json.JSONObject;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.ResponseHandler;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;
import com.thoughtworks.xstream.XStream;
import com.thoughtworks.xstream.core.util.QuickWriter;
import com.thoughtworks.xstream.io.HierarchicalStreamWriter;
```

```

import com.thoughtworks.xstream.io.xml.PrettyPrintWriter;
import com.thoughtworks.xstream.io.xml.XppDriver;
import myf.caption10.vo.resp.Article;
import myf.caption10.vo.resp.NewsMessage;
import myf.caption10.vo.resp.TextMessage;
/**
 * @author 牟云飞
 * @tel 1556257xxxx
 * @QQ 1147417467
 * @java Job 2012 年
 * @时间 2014-7-22
 */
public class WxUtil {
    /**-----*/
    /*******主动发送消息时不需要消息体进行加密，格式为 JSON 格式******/
    /**-----*/
    //微信企业号唯一标识 CorpID
    public static final String MESSAGE_CORPID="6666666666";
    //主动调用：管理组凭证密钥
    public static final String MESSAGE_SECRET="9999999999999999";
    //主动调用：发送消息获得 token
    public static String access_token;
    //主动调用：请求 token 的时间
    public static Date access_token_date;
    //主动调用：发送消息获得 token
    public static String jsapi_ticket;
    //主动调用：请求 token 的时间
    public static Date jsapi_ticket_date;
    //微信会话：发送消息获得 token
    public static String taking_access_token;
    //微信会话：请求 token 的时间
    public static Date taking_access_token_date;
    //微信会话：凭证密钥
    public static final String TAKING_MESSAGE_SECRET="888888888888";
    /**-----*/
    /****被动响应消息需要对消息体，先进行 AES 加密，处理后再 base64 加密,格式为 XML 格式**/
    /**-----*/
    //被动响应消息配置：token
    public static final String RESP_MESSAGE_TOKEN="muyunfeimyf";
    //被动响应消息配置：AES 密钥。密钥生成规则：32 位的明文经过 base64 加密后，去掉末尾
    // =号，形成 43 位的密钥
    public static final String RESP_MESSAGE_ENCODINGAESKEY=
    "bXV5dW5mZWl3ZWl4aW5rYWlmYWppYW9jaGVuZ215Zm0";

    /**-----*/
    /*******消息类型******/

```

```

/**-----*/
//消息类型: 文本
public static final String MESSAGE_TYPE_TEXT = "text";
public static final String MESSAGE_TYPE_NEWS = "news";
/**-----*/
/*****消息事件类型*****/
/**-----*/
//消息类型: 事件推送
public static final String MESSAGE_TYPE_EVENT = "event";
public static final String EVENT_TYPE_ENTERAGENT = "enter_agent";
/**-----*/
/*****微信应用 ID*****/
/**-----*/
public static final String AGENT_ID_ASSIST = "0";//默认应用“企业小助手”
public static final String AGENT_ID_HELLOWORD = "5";//企业资讯应用 ID
public static final String AGENT_ID_REGISTER="14";//微信考勤应用 ID
/**-----*/
/*****其他配置信息*****/
/**-----*/
//外网域名
public static String webUrl="http://myfmyfmyfmyf.vicp.cc/wx_demo";

/**
 * 扩展 xstream, 使其支持 CDATA
 */
private static XStream xstream = new XStream(new XppDriver() {
    public HierarchicalStreamWriter createWriter(Writer out) {
        return new PrettyPrintWriter(out) {
            // 对所有 xml 节点的转换都增加 CDATA 标记
            boolean cdata = true;

            @SuppressWarnings("unchecked")
            public void startNode(String name, Class clazz) {
                super.startNode(name, clazz);
            }

            protected void writeText(QuickWriter writer, String text) {
                if (cdata) {
                    writer.write("<![CDATA[");
                    writer.write(text);
                    writer.write("]]>");
                } else {
                    writer.write(text);
                }
            }
        };
    }
});

```

```

    };

    }

});

/**
 * 文本消息对象转换成 XML
 *
 * @param textMessage 文本消息对象
 * @return xml
 */
public static String messageToXml(TextMessage textMessage) {
    xstream.alias("xml", textMessage.getClass());
    return xstream.toXML(textMessage);
}

/**
 * 图文消息对象转换成 XML
 *
 * @param newsMessage 图文消息对象
 * @return xml
 */
public static String messageToXml(NewsMessage newsMessage) {
    xstream.alias("xml", newsMessage.getClass());
    xstream.alias("item", new Article().getClass());
    return xstream.toXML(newsMessage);
}

/**
 * 从微信获得 access_token
 * @return
 */
public String getTokenFromWx(){
    //微信企业号标识
    String corpid=MESSAGE_CORPID;
    //管理组凭证密钥
    String corpsecret=MESSAGE_SECRET;
    //获取的标识
    String token="";
    //1.判断 access_token 是否存在，不存在则直接申请
    //2.判断时间是否过期，过期 (>=7200 秒) 申请，否则直接返回以后的 token
    if(null==access_token||"".equals(access_token)||(new
Date().getTime()-access_token_date.getTime())>=(7000*1000)){
        CloseableHttpClient httpClient = HttpClients.createDefault();
        try {
            //利用 GET 形式获得 token
            HttpGet httpget = new HttpGet("https://qyapi.weixin.qq.com/cgi-bin/gettoken?corpid="+corpid+"&corpsecret="+corpsecret);
            //Create a custom response handler

```



```

        ResponseHandler<JSONObject> responseHandler = new
ResponseHandler<JSONObject>() {

            public JSONObject handleResponse(
                final HttpResponse response) throws
ClientProtocolException, IOException {
                int status = response.getStatusLine().getStatusCode();
                if (status >= 200 && status < 300) {
                    HttpEntity entity = response.getEntity();
                    if (null!=entity) {
                        String result= EntityUtils.toString(entity);
                        //根据字符串生成 JSON 对象
                        JSONObject resultObj = JSONObject.fromObject
(result);

                        return resultObj;
                    }else{
                        return null;
                    }
                } else {
                    throw new ClientProtocolException("Unexpected response
status: " + status);
                }
            }

        };
        //返回的 JSON 对象
        JSONObject responseBody = httpClient.execute(httpget,
responseHandler);
        if (null!=responseBody) {
            token= (String) responseBody.get("access_token");
            //返回 Token
        }

        httpClient.close();

        //设置全局变量
        access_token=token;
        access_token_date=new Date();
    }catch (Exception e) {
        e.printStackTrace();
    }
    }else{
        token=access_token;
    }
    return token;
}
}

```

```

/**
 * 从微信获得 jsapi_ticket
 * @return
 */
public String getJsapiTicketFromWx(){
    String token=getTokenFromWx();//token
    //1.判断 jsapi_ticket 是否存在,不存在直接申请
    //2.判断时间是否过期,过期(>=7200 秒)申请,否则直接返回以后的 Token
    if(null==jsapi_ticket||"".equals(jsapi_ticket)|| (new
Date().getTime()-jsapi_ticket_date.getTime())>=(7000*1000)){

        CloseableHttpClient httpClient = HttpClients.createDefault();
        try {
            //利用 get 形式获得 token
            HttpGet httpget = new HttpGet("https://qyapi.weixin.qq.com/
cgi-bin/get_jsapi_ticket?access_token="+token);
            // Create a custom response handler
            ResponseHandler<JSONObject> responseHandler = new
ResponseHandler<JSONObject>() {

                public JSONObject handleResponse(
                    final HttpResponse response) throws
ClientProtocolException, IOException {
                    int status = response.getStatusLine().getStatusCode();
                    if (status >= 200 && status < 300) {
                        HttpEntity entity = response.getEntity();
                        if(null!=entity){
                            String result= EntityUtils.toString(entity);
                            //根据字符串生成 JSON 对象
                            JSONObject resultObj = JSONObject.fromObject
(result);

                            return resultObj;
                        }else{
                            return null;
                        }
                    } else {
                        throw new ClientProtocolException("Unexpected
response status: " + status);
                    }
                }
            };
            //返回的 JSON 对象
            JSONObject responseBody = httpClient.execute(httpget,
responseHandler);
            if(null!=responseBody){

```

```
        jsapi_ticket= (String) responseBody.get("ticket");//返回
token
    }
    jsapi_ticket_date=new Date();
    httpclient.close();
} catch (Exception e) {
    e.printStackTrace();
}
}
return jsapi_ticket;
}
/****微信 JS 签名*****/

public static Map<String, String> sign(String jsapi_ticket, String url) {
    Map<String, String> ret = new HashMap<String, String>();
    String nonce_str = create_nonce_str();
    String timestamp = create_timestamp();
    String string1;
    String signature = "";

    //注意, 这里参数名必须全部小写, 且必须有序
    string1 = "jsapi_ticket=" + jsapi_ticket +
        "&noncestr=" + nonce_str +
        "&timestamp=" + timestamp +
        "&url=" + url;

    try
    {
        MessageDigest crypt = MessageDigest.getInstance("SHA-1");
        crypt.reset();
        crypt.update(string1.getBytes("UTF-8"));
        signature = byteToHex(crypt.digest());
    }
    catch (NoSuchAlgorithmException e)
    {
        e.printStackTrace();
    }
    catch (UnsupportedEncodingException e)
    {
        e.printStackTrace();
    }

    ret.put("url", url);
    ret.put("jsapi_ticket", jsapi_ticket);
    ret.put("nonceStr", nonce_str);
    ret.put("timestamp", timestamp);
    ret.put("signature", signature);
}
```

```

        return ret;
    }

    private static String byteToHex(final byte[] hash) {
        Formatter formatter = new Formatter();
        for (byte b : hash)
        {
            formatter.format("%02x", b);
        }
        String result = formatter.toString();
        formatter.close();
        return result;
    }

    private static String create_nonce_str() {
        return UUID.randomUUID().toString();
    }

    private static String create_timestamp() {
        return Long.toString(System.currentTimeMillis() / 1000);
    }

    /**
     * 根据 code 获得人员
     */
    public String getUserIdByCode(String code) {
        // 获取微信号
        String token = getTokenFromWx();
        try {
            CloseableHttpClient httpClient = HttpClients.createDefault();
            HttpPost httpPost = new HttpPost("https://qyapi.weixin.qq.com/cgi-bin/user/getuserinfo?access_token="+token+"&code="+code+"&agentid="+AGENT_ID_REGISTER);
            // Create a custom response handler
            ResponseHandler<JSONObject> responseHandler = new ResponseHandler<JSONObject>() {

                public JSONObject handleResponse(
                    final HttpResponse response) throws ClientProtocolException,
                    IOException {

                    int status = response.getStatusLine().getStatusCode();
                    if (status >= 200 && status < 300) {
                        HttpEntity entity = response.getEntity();
                        if (null != entity) {
                            String result = EntityUtils.toString(entity);

```



```

        //根据字符串生成 JSON 对象
        JSONObject resultObj = JSONObject.fromObject
(result);

        return resultObj;
    }else{
        return null;
    }
} else {
    throw new ClientProtocolException("Unexpected response
status: " + status);
}
}

};
//返回的 JSON 对象
JSONObject responseBody = httpClient.execute(httpPost,
responseHandler);
//AppLogUtil.getAppLogger().info(responseBody.toString());
if(null==responseBody.getString("UserId")){
    return null;
}else{
    return responseBody.getString("UserId");
}
}catch (Exception e) {
    //e.printStackTrace();
    return null;
}
}
}
}

```

11.3.2 编写回调服务

开启回调服务，创建 Servlet 回调服务类 RegisterMobileServlet.java，用于开启回调服务，接收回调事件等，示例代码如下：

```

package myf.caption11.servlet;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import myf.caption10.QQTool.WXBizMsgCrypt;
import myf.caption10.util.WxUtil;
import myf.caption11.service.impl.RegisterMobileServiceImpl;
import myf.caption11.service.interf.IRegisterMobileService;

```

```

/**
 * 微信考勤回调地址
 * @author 牟云飞
 * @tel 1556257xxxx
 * @QQ 1147417467
 * @java Job 2012 年
 * @时间 2014-7-22
 */
public class RegisterMobileServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        //微信加密签名
        String signature = request.getParameter("msg_signature");
        //System.out.println("signature:"+signature);
        //时间戳
        String timestamp = request.getParameter("timestamp");
        //System.out.println("timestamp:"+timestamp);
        //随机数
        String nonce = request.getParameter("nonce");
        //System.out.println("nonce:"+nonce);
        //随机字符串
        String echostr = request.getParameter("echostr");
        //System.out.println("echostr:"+echostr);
        String sToken = WxUtil.RESP_MESSAGE_TOKEN;
        String sCorpID = WxUtil.MESSAGE_CORPID;
        String sEncodingAESKey = WxUtil.RESP_MESSAGE_ENCODINGAESKEY;
        try {
            WXBizMsgCrypt wxcpt = new WXBizMsgCrypt(sToken, sEncodingAESKey,
sCorpID);

            String sEchoStr; //需要返回的明文
            sEchoStr = wxcpt.VerifyURL(signature, timestamp,
                nonce, echostr);
            System.out.println("verifyurl echostr: " + sEchoStr);
            //验证 URL 成功, 将 sEchoStr 返回
            PrintWriter out = response.getWriter();
            out.write(sEchoStr);
            out.flush();
            out.close();
        } catch (Exception e) {
            //验证 URL 失败, 错误原因请查看异常
            e.printStackTrace();
        }
    }
}

```

```
protected void doPost(HttpServletRequest request, HttpServletResponse
response)
    throws ServletException, IOException {
    //读取消息, 执行消息处理
    IRegisterMobileService service = (IRegisterMobileService) new
RegisterMobileServiceImpl();
    service.receiveMsg(request, response);
}
}
```

为了使服务器能够识别 servlet 服务, 我们需要在 web.xml 中增加 servlet 服务配置项, 示例代码如下:

```
<!-- 微信考勤应用的请求类, 回调模式使用-->
<servlet>
    <servlet-name>registerMobileServlet</servlet-name>
    <servlet-class>
        myf.caption11.servlet.RegisterMobileServlet
    </servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>registerMobileServlet</servlet-name>
    <url-pattern>/registerMobileServlet.slt</url-pattern>
</servlet-mapping>
```

11.3.3 考勤信息实体类

vo, 值对象, 在 vo 中创建考勤信息类 AttendanceInfo, 示例代码如下:

```
package myf.caption11.vo;

import java.io.Serializable;
import java.util.Date;
/**
 * 考勤信息类
 *
 * @author 牟云飞
 *
 * <p>Modification History:</p>
 * <p>Date          Author          Description</p>
 * <p>-----<-----</p>
 * <p>2015-05-15      牟云飞          new          </p>
 */
public class AttendanceInfo implements Serializable {

    private static final long serialVersionUID = 1L;
```

```

private String userId; //员工主键
private Date day; //考勤日期
private long dayStatus; //状态

public Date getDay() {
    return day;
}
public void setDay(Date day) {
    this.day = day;
}
public long getDayStatus() {
    return dayStatus;
}
public void setDayStatus(long dayStatus) {
    this.dayStatus = dayStatus;
}
public String getUserId() {
    return userId;
}
public void setUserId(String userId) {
    this.userId = userId;
}
}

```

11.3.4 创建业务层服务类

server 层通过接口实现的方式，用于实现业务与数据、UI 的分离。创建接口类 `IRegisterMobileService.java` 和实现类 `RegisterMobileServiceImpl.java`，接口类示例代码如下：

```

package myf.caption11.service.interf;

import java.util.Date;
import java.util.List;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import myf.caption11.vo.AttendanceInfo;
/**
 *
 * 移动考勤 service
 * @tel      1556257xxxx
 * @QQ      1147417467
 * <p>Modification History:</p>
 * <p>Date      Author      Description</p>
 * <p>-----</p>
 * <p>May 15, 2015      牟云飞      新建</p>
 */

```



```
public interface IRegisterMobileService {

    /**
     * 参数为员工账号和考勤月份（格式为考勤月份首日：2015-05-01），返回员工指定月份的
     所有考勤信息
     * @param rtxCode
     * @param date
     * @return
     */
    public List<AttendanceInfo> retrieveFromHr(String empCode ,Date date);

    /**
     * 回调模式处理消息
     * @param request
     * @param response
     * @return
     */
    public String receiveMsg(HttpServletRequest request, HttpServletResponse
response);
}
```

接口实现类 RegisterMobileServiceImpl 示例代码如下：

```
package myf.caption11.service.impl;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.io.StringReader;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import javax.servlet.ServletInputStream;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import myf.caption10.QQTool.WxBizMsgCrypt;
import myf.caption10.dao.impl.WxEventDaoImpl;
import myf.caption10.dao.interf.IWxEventDao;
import myf.caption10.po.WxEvent;
import myf.caption10.util.WxUtil;
import myf.caption10.vo.resp.Article;
import myf.caption10.vo.resp.NewsMessage;
import myf.caption10.vo.resp.TextMessage;
import myf.caption11.service.interf.IRegisterMobileService;
```

```

import myf.caption11.vo.AttendanceInfo;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;

/**
 * @author 牟云飞
 * @tel 1556257xxxx
 * @QQ 1147417467
 * @java Job 2012 年
 * @时间 2014-7-22
 */
public class RegisterMobileServiceImpl implements IRegisterMobileService {
    private IWxEventDao eventDao = new WxEventDaoImpl();

    //回调模式处理消息
    public String receiveMsg(HttpServletRequest request, HttpServletResponse
response) {
        //微信加密签名
        String sReqMsgSig = request.getParameter("msg_signature");
        //时间戳
        String sReqTimeStamp = request.getParameter("timestamp");
        //随机数
        String sReqNonce = request.getParameter("nonce");
        String sToken = WxUtil.RESP_MESSAGE_TOKEN;
        String sCorpID = WxUtil.MESSAGE_CORPID;
        String sEncodingAESKey = WxUtil.RESP_MESSAGE_ENCODINGAESKEY;
        try {
            //POST 请求的密文数据
            //sReqData = HttpUtils.PostData();
            ServletInputStream in = request.getInputStream();
            BufferedReader reader = new BufferedReader(new InputStreamReader(in));
            String sReqData = "";
            String itemStr = ""; //作为输出字符串的临时串, 用于判断是否读取完毕
            while (null != (itemStr = reader.readLine())) {
                sReqData += itemStr;
            }
            //对消息进行处理, 获得明文
            WXBizMsgCrypt wxcpt = new WXBizMsgCrypt(sToken, sEncodingAESKey,
sCorpID);
            String sMsg = wxcpt.DecryptMsg(sReqMsgSig, sReqTimeStamp, sReqNonce,
sReqData);
            //输出解密后的文件
            //For example:
            DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();

```

```

        DocumentBuilder db = dbf.newDocumentBuilder();
        StringReader sr = new StringReader(sMsg);
        InputSource is = new InputSource(sr);
        Document document = db.parse(is);
        Element root = document.getDocumentElement();
        //判断类型
        NodeList nodelist_msgType = root.getElementsByTagName("MsgType");
        String recieveMsgType = nodelist_msgType.item(0).getTextContent();
        String content="";
        String sEncryptMsg="";//加密的消息
        if("text".equals(recieveMsgType)){
            //如果是文本消息.....
            //获得内容
            NodeList nodelist1 = root.getElementsByTagName("Content");
            String resContent = nodelist1.item(0).getTextContent();
            //哪个员工进入
            NodeList enter_people_note = root.getElementsByTagName
("FromUserName");
            String enter_people = enter_people_note.item(0).getTextContent();
            //发送HelloWorld消息
            TextMessage text = new TextMessage();
            text.setMsgType(WxUtil.MESSAGE_TYPE_TEXT);
            //发送者
            text.setFromUserName(WxUtil.MESSAGE_CORPID);
            //接收者
            text.setToUserName(enter_people);
            long time=new Date().getTime();
            text.setCreateTime(time);
            text.setContent("接收到"+enter_people+"信息: "+resContent);
            String sRespData=WxUtil.messageToXml(text);
            //响应加密消息
            content=sEncryptMsg= wxcpt.EncryptMsg(sRespData, time+"", sReqNonce);
            //保存数据库
            //.....
        }else if("event".equals(recieveMsgType)){//如果是事件消息
            //获得事件类型
            NodeList nodelist1 = root.getElementsByTagName("Event");
            String event_type = nodelist1.item(0).getTextContent();
            if("enter_agent".equals(event_type)){//进入应用的事件
                //哪个员工进入
                NodeList enter_people_note = root.getElementsByTagName
("FromUserName");
                String enter_people = enter_people_note.item(0).getTextContent();

                //判断是否第一次进入应用, 若是第一次进入应用, 则推送一条消息
                List<WxEvent> enterList = eventDao.queryList(enter_people,

```



```

        Integer.valueOf(WxUtil.AGENT_ID_HELLOWORD)
        ,WxUtil.EVENT_TYPE_ENTERAGENT);
    if(0==enterList.size()){
        //第一次进入，则推送一条消息
        //发送 HelloWord 消息
        TextMessage text = new TextMessage();
        text.setMsgType(WxUtil.MESSAGE_TYPE_TEXT);
        //发送者
        text.setFromUserName(WxUtil.MESSAGE_CORPID);
        //接收者
        text.setToUserName(enter_people);
        long time=new Date().getTime();
        text.setCreateTime(time);
        text.setContent(enter_people+" HelloWord!!");
        String sRespData=WxUtil.messageToXml(text);
        //响应加密消息
        content=sEncryptMsg= wxcpt.EncryptMsg(sRespData, time+"",
sReqNonce);
    }
    //保存日志至数据库
    try{
        WxEvent event = new WxEvent();
        event.setEventType("enter_agent");
        event.setFromUser(enter_people);
        event.setCreateTime(new Timestamp(new Date().getTime()));
        event.setAgentId(Integer.valueOf(WxUtil.AGENT_ID_
HELLOWORD));

        event.setToUser(WxUtil.MESSAGE_CORPID);
        eventDao.add(event);
    }catch(Exception e){
        System.out.println("移动考勤进入应用，数据库保存失败");
        e.printStackTrace();
    }
}
else if("click".equals(event_type)){
    //获得 event_key
    NodeList node_EventKey = root.getElementsByTagName("EventKey");
    String eventKey = node_EventKey.item(0).getTextContent();
    if("KEY_HELLO".equals(eventKey)){
        //发送 news 消息
        NewsMessage news=new NewsMessage();
        news.setFromUserName(WxUtil.MESSAGE_CORPID);//消息来源
        news.setMsgType(WxUtil.MESSAGE_TYPE_NEWS);
        NodeList enter_people_note = root.getElementsByTagName
("FromUserName");
        String enter_people=enter_people_note.item(0).getTextContent();

```



```

        news.setToUserName(enter_people);//回复人
        long time=new Date().getTime();
        news.setCreateTime(time);//创建时间
        news.setArticleCount(1);//文章数量
        List<Article> articles = new ArrayList<Article>();
        Article article=new Article();
        article.setTitle("使用帮助");
        article.setPicUrl(WxUtil.webUrl+"/image/info.png");
        article.setUrl(WxUtil.webUrl+"/help.html");
        article.setDescription("HELLOWORD-使用帮助,作者: 牟云飞,
Tel:1556257xxxx.....");
        articles.add(article);
        news.setArticles(articles);
        String sRespData=WxUtil.messageToXml(news);
        //响应加密消息
        content=sEncryptMsg = wxcpt.EncryptMsg(sRespData, time+"",
sReqNonce);
    }
}
}
//!!!!!!!!!!!!!!!!!!!!!!设置回复!!!!!!!!!!!!!!
//-----
if(null!=content&&!"".equals(content)){
    //输出
    PrintWriter out = response.getWriter();
    out.write(sEncryptMsg);
    out.flush();
    out.close();
}
} catch (Exception e) {
    // TODO
    // 解密失败,失败原因请查看异常
    e.printStackTrace();
}
return "";
}

//查询考勤信息
public List<AttendanceInfo> retrieveFromHr(String empCode, Date date) {
    //本方法体内为测试数据,读者自行编写所需方法体
    List<AttendanceInfo> list=new ArrayList<AttendanceInfo>();
    for (int i = 0; i < 16; i++) {
        AttendanceInfo info = new AttendanceInfo();
        info.setDay(new Date(date.getTime()+24L*60L*60L*1000L*Long.
valueOf(i)));
        info.setUserId(empCode);
    }
}

```

```

        info.setDayStatus(0);
        if(i==13||i==14)
            info.setDayStatus(2);
        if(i==11)
            info.setDayStatus(4);
        list.add(info);
    }
    return list;
}

public IWxEventDao getEventDao() {
    return eventDao;
}

public void setEventDao(IWxEventDao eventDao) {
    this.eventDao = eventDao;
}
}

```

11.3.5 服务跳转类

Action 跳转类 RegisterMobileAction.java，基于 Struts 中的 DispatchAction 类实现，通过配置 Web.xml 增加 Struts 配置，使其识别并捕获处理.do 服务，代码如下：

```

<!-- struts 配置-->
<servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
    <init-param>
        <param-name>config</param-name>
        <param-value>/WEB-INF/struts-config.xml</param-value>
    </init-param>
    <init-param>
        <param-name>debug</param-name>
        <param-value>3</param-value>
    </init-param>
    <init-param>
        <param-name>detail</param-name>
        <param-value>3</param-value>
    </init-param>
    <load-on-startup>0</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
</servlet-mapping>

```

通过浏览器拦截.do 服务后，匹配 struts-config.xml 文件中的 action 服务，识别并进入相应的 Java 服务实现类。struts-config.xml 配置内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts
Configuration 1.3//EN" "http://struts.apache.org/dtds/struts-config_1_3.dtd">

<struts-config>
  <form-beans >
    <form-bean name="strutsOneForm" type="org.apache.struts.action.
DynaActionForm" >
      <form-property name="userName" type="java.lang.String"></form-property>
      <form-property name="pwd" type="java.lang.String"></form-property>
    </form-bean>
  </form-beans>
  <global-exceptions />
  <global-forwards />
  <action-mappings >
    <action
      attribute="strutsOneForm"
      name="strutsOneForm"
      input="/index.jsp"
      parameter="action"
      path="/registerMobileAction"
      scope="request"
      type="myf.caption11.action.RegisterMobileAction"
      cancellable="true">
      <forward name="register" path="/registerMobile.jsp" />
      <forward name="succ" path="/registerSuccess.jsp" />
      <forward name="fail" path="/registerFail.jsp" />
      <forward name="safePage" path="/safePage.jsp" />
      <forward name="logPage" path="/registerInfoQuery.jsp" />
    </action>
  </action-mappings>
</struts-config>
```

action 服务处理类 RegisterMobileAction.java 示例代码如下：

```
package myf.caption11.action;

import java.sql.Timestamp;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;
import java.util.Map;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```

import myf.caption10.util.WxUtil;
import myf.caption11.service.impl.RegisterMobileServiceImpl;
import myf.caption11.service.interf.IRegisterMobileService;
import myf.caption11.vo.AttendanceInfo;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.actions.DispatchAction;
/**
 * 微信考勤 Action
 * @author 牟云飞
 * @tel 1556257xxxx
 * @QQ 1147417467
 * @java Job 2012 年
 * @时间 2014-7-22
 */
public class RegisterMobileAction extends DispatchAction {

    private IRegisterMobileService registerMobileServiceImpl=
new RegisterMobileServiceImpl();

    //初始化考勤页面
    public ActionForward initPage(ActionMapping mapping, ActionForm form,
        HttpServletRequest req, HttpServletResponse response){
        //识别微信浏览器
        String userAgent=req.getHeader("User-Agent");//里面包含了设备类型
        if(-1==userAgent.indexOf("MicroMessenger")){
            //如果不是微信浏览器,则跳转到安全页
            return mapping.findForward("safePage");
        }
        //生成微信 js 授权
        WxUtil msgUtil=new WxUtil();
        String jsapi_ticket=msgUtil.getJsapiTicketFromWx();//签名
        String url = WxUtil.webUrl //域名地址, 自己的外网请求的域名, 如:
//http://www.baidu.com/Demo
        + req.getServletPath() //请求页面或其他地址
        + "?" + (req.getQueryString()); //参数
        Map<String, String> ret = WxUtil.sign(jsapi_ticket, url);
        req.setAttribute("str1", ret.get("signature"));
        req.setAttribute("time", ret.get("timestamp"));
        req.setAttribute("nonceStr", ret.get("nonceStr"));

        //设置公司的腾讯地图坐标、地址 http://lbs.qq.com/javascript_v2/case-
//run.html#sample-map-coordinate
        req.setAttribute("hyLatitude", "39.91026");//纬度
        req.setAttribute("hyLongitude", "116.39946");//经度
    }
}

```



```

        req.setAttribute("hyCircle", "550");//范围
        //获取网址
        req.setAttribute("addressUrl", WxUtil.webUrl);//范围
        return mapping.findForward("register");
    }

    //打卡
    public ActionForward register(ActionMapping mapping, ActionForm form,
        HttpServletRequest req, HttpServletResponse resp){
        //获得人员当前的位置
        String positionX=req.getParameter("posX")+"";//纬度
        String positionY=req.getParameter("posY")+"";//经度
        //获得 OAuth 2 验证
        String code=req.getParameter("code");
        String state=req.getParameter("state");
        //验证 checkid 是否在范围内, 不在范围内则为恶意刷新
        //        long checkId=0;
        //        if(null==req.getParameter("checkCode")||"".equals
        (req.getParameter("checkCode"))||Long.valueOf(req.getParameter("checkCode"))
        >MessageUtil.regesiterCurMaxCode){
            //                return "safePage";
            //        }
            //        checkId=Long.valueOf(req.getParameter("checkCode"));
            //        //判断是否在数据库中, 存在为恶意刷新
            //        WxCheckLog isEixtData = wxCheckLogServiceImpl.retrieveById(checkId);
            //        if(null!=isEixtData){
            //            req.setAttribute("curUser", isEixtData.getUserId());
            //            return "safePage";
            //        }
            //生成微信 js 授权
            WxUtil msgUtil=new WxUtil();
            String jsapi_ticket=msgUtil.getJsapiTicketFromWx();//签名
            String url = WxUtil.webUrl //域名地址, 自己的外网请求的域名, 如:
            //http://www.baidu.com/Demo
                + req.getServletPath() //请求页面或其他地址
                + "?" + (req.getQueryString()); //参数
            Map<String, String> ret = WxUtil.sign(jsapi_ticket, url);
            req.setAttribute("str1", ret.get("signature"));
            req.setAttribute("time", ret.get("timestamp"));
            req.setAttribute("nonceStr", ret.get("nonceStr"));
            if("null".equals(positionX)||"".equals(positionX)||"null".equals
            (positionY)||"".equals(positionY)){
                return mapping.findForward("fail");
            }
            //根据 code 获得人员
            WxUtil util=new WxUtil();

```

```

String userId=util.getUserIdByCode(code);
if(null==userId){
    //如果获取信息失败，返回打卡失败页面
    return mapping.findForward("fail");
}
req.setAttribute("userId", userId);//范围
// 判断人员是否数据本人，不是则为恶意刷新
// if(!userId.equals(isEixtData.getUserId())){
//     return "safePage";
// }
// 判断是否存在地理位置，原数据存在地理位置则为恶意刷新
// if(null!=isEixtData.getLocationX() && null!=isEixtData.getLocationY()){
//     return "safePage";
// }
//调用 hessian 服务
try {
    //打卡时间
    Timestamp time = new Timestamp(new Date().getTime());
    SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    System.out.println("本机时间: "+ format.format(time));
    /*****调用 HR 系统服务方法，推送打卡数据*****/
    boolean hrResult=true;//读者自行调用自己的服务
    /*****end,调用 HR 系统服务方法，推送打卡数据*****/
    if(true==hrResult){
        System.out.println(userId+" 推送数据考勤成功");
        return mapping.findForward("succ");
    }else{
        return mapping.findForward("fail");
    }
} catch (Exception e) {
    e.printStackTrace();
    return mapping.findForward("fail");
}
}

/**
 * 坐标转换，腾讯地图坐标转换成百度地图坐标
 * @param lat 腾讯地图坐标纬度
 * @param lon 腾讯地图坐标经度
 * @return 返回结果：经度，纬度
 */
public String map_tx2bd(double lat, double lon){
    double bd_lat;
    double bd_lon;
    double x_pi=3.14159265358979324;
    double x = lon, y = lat;

```

```

        double z = Math.sqrt(x * x + y * y) + 0.00002 * Math.sin(y * x_pi);
        double theta = Math.atan2(y, x) + 0.000003 * Math.cos(x * x_pi);
        bd_lon = z * Math.cos(theta) + 0.0065;
        bd_lat = z * Math.sin(theta) + 0.006;
        return bd_lon+","+bd_lat;
    }

    /**
     * 坐标转换, 百度地图坐标转换成腾讯地图坐标
     * @param lat 百度地图坐标纬度
     * @param lon 百度地图坐标经度
     * @return 返回结果: 纬度, 经度
     */
    public String map_bd2tx(double lat, double lon){
        double tx_lat;
        double tx_lon;
        double x_pi=3.14159265358979324;
        double x = lon - 0.0065, y = lat - 0.006;
        double z = Math.sqrt(x * x + y * y) - 0.00002 * Math.sin(y * x_pi);
        double theta = Math.atan2(y, x) - 0.000003 * Math.cos(x * x_pi);
        tx_lon = z * Math.cos(theta);
        tx_lat = z * Math.sin(theta);
        return tx_lat+","+tx_lon;
    }

    //获取打卡日志
    public ActionForward retrieveRecord(ActionMapping mapping, ActionForm form,
        HttpServletRequest req, HttpServletResponse resp){
        //生成微信 js 授权
        WxUtil util=new WxUtil();
        String jsapi_ticket=util.getJsapiTicketFromWx();//签名
        String url = WxUtil.webUrl //域名地址, 自己的外网请求的域名, 如:
        //http://www.baidu.com/Demo
        + req.getServletPath() //请求页面或其他地址
        + "?" + (req.getQueryString()); //参数
        Map<String, String> ret = WxUtil.sign(jsapi_ticket, url);
        req.setAttribute("str1", ret.get("signature"));
        req.setAttribute("time", ret.get("timestamp"));
        req.setAttribute("nonceStr", ret.get("nonceStr"));
        //获得 OAuth2 验证
        String code=req.getParameter("code");
        String state=req.getParameter("state");
        //根据 code 获得人员
        String userId=util.getUserIdByCode(code);
        System.out.println(userId+" 查询考勤日志");
        if(null!=userId){

```



```

//格式为考勤月份首日：2015-05-01
Timestamp curDate = new Timestamp(new Date().getTime());
SimpleDateFormat sd = new SimpleDateFormat("yyyy-MM-01 00:00:00");
Date date = Timestamp.valueOf(sd.format(curDate));
List<AttendanceInfo> list = registerMobileServiceImpl.retrieveFromHr
(userId, date);
//整理数据格式
StringBuffer regInfoStr=new StringBuffer("");
for (int i = 0; i < list.size(); i++) {
    AttendanceInfo info = list.get(i);
    regInfoStr.append(info.getDay().getDate()+" "+info.getDayStatus());
    if(i!=(list.size()-1)){
        regInfoStr.append(";");
    }
}
req.setAttribute("regInfoStr", regInfoStr);
//将当前人传到页面
req.setAttribute("userId", userId.toString());
//System.out.println(regInfoStr);
return mapping.findForward("logPage");
}
return mapping.findForward("safePage");
}
}

```

11.3.6 JSP 考勤打卡 Map 页

员工通过点击微信菜单“考勤打卡”进入打卡页面 `registerMobile.jsp`，初始化定位员工当前位置以及企业位置，在一定范围内，员工能够通过微信进行考勤打卡。对于超出范围的员工，则需通过重新定位，进入范围内方可打卡，示例代码如下：

```

<%@ page language="java" contentType="text/html; charset=GBK"
    pageEncoding="GBK"%>
<!-- @author 牟云飞 -->
<html>
<head>
<title>移动考勤</title>
<meta name="viewport" content="initial-scale=1.0, user-scalable=no"/>
<meta name="viewport" content="width=device-width, initial-scale=1">
<script type="text/javascript" src="<%=request.getContextPath()%>/js/
jquery-1.7.js"></script>
<script type="text/javascript" src="http://res.wx.qq.com/open/js/jweixin-
1.0.0.js"></script>
<script charset="utf-8" src="http://map.qq.com/api/js?v=2.exp&libraries=
convertor&key=QWZBZ-JHR3R-IN4W2-WCRIW-AOTYV-DUBE7"></script>
<style type="text/css">

```



```

*{
    margin:0px;
    padding:0px;
}
body, button, input, select, textarea {
    font: 12px/16px Verdana, Helvetica, Arial, sans-serif;
}
p{
    width:603px;
    padding-top:3px;
    margin-top:10px;
    overflow:hidden;
}
</style>
<script>
    var searchService,map,markers = [];
    var init = function() {
        var cpLatitude=$("#hyLatitude").val();
        var cpLongitude=$("#hyLongitude").val();
        var cpCircle=parseInt($("#hyCircle").val());
        var center = new qq.maps.LatLng(cpLatitude,cpLongitude);
        map = new qq.maps.Map(document.getElementById('container'),{
            center: center,
            zoom: 15
        });
        new qq.maps.Circle({
            center:new qq.maps.LatLng(cpLatitude,cpLongitude),
            radius: cpCircle,
            map: map
        });
        //知识锚点
        var marker = new qq.maps.Marker({
            position: center,
            map: map
        });
        //设置当前位置
        var anchor = new qq.maps.Point(0, 0);
        var size = new qq.maps.Size(31, 34);
        var markerIcon = new qq.maps.MarkerImage(
            "<%=request.getContextPath()%>/image/target1.png",
            size,null,null,size
        );
        marker.setIcon(markerIcon);
    }

    wx.config({

```

```

    debug : false, //开启调试模式,调用的所有的 API 返回值会在客户端弹出消息提示,
//输出接口信息,若要查看传入的参数,可以在 PC 端打开,参数信息会通过 log 打印出来,仅在 PC
//端时才能打印
    appId : 'wx0c90608d772d2164', //必填,企业号的唯一标识,此处填写企业号 CorpID
    timestamp : "${time}", //必填,生成签名的时间戳
    nonceStr : '${nonceStr}', //必填,生成签名的随机串
    signature : '${str1}', // 必填, 签名
    jsApiList : ['hideOptionsMenu','getLocation','checkJsApi']//必填,需
//要使用的 JS 接口列表
  });

  wx.ready(function(){
    //config 信息验证后会执行 ready 方法,所有接口调用都必须在 config 接口获得结
//果之后.config 是一个客户端的异步操作,如果需要在页面加载时就调用相关接口,则必须把相关
//接口放在 ready 函数中调用,以确保正确执行.对于用户触发时才调用的接口,则可以直接调用,
//不需要放在 ready 函数中.
    wx.hideOptionsMenu();
    locationAgain();//获取位置
  });

  //打卡
  function registerClick(){
    if(null==$('#positionX').val()||""==$('#positionX').val()||null==
$('#positionY').val()||""==$('#positionY').val()){
      alert("正在获取当前位置,请稍后打卡");
      return;
    }
    var cpCircle=parseInt($("#hyCircle").val()+30;
    var distance= getFlatternDistance(parseFloat($("#positionX").
val()),parseFloat($("#positionY").val()));
    if(distance>=cpCircle){
      alert("尚未在打卡范围内,请靠近打卡范围后重新定位");
      return;
    }
    if($("#alreadyClick").val()=="false" || $("#alreadyClick").val()==
false || $("#alreadyClick").val()==null){
      var getUrl=$("#addressUrl").val()+"/registerMobileAction.do?action=
register&posX="+$('#positionX').val()+"&posY="+$('#positionY').val()+"&check
Code="+$('#checkInfo').val();
      var changeurl=getUrl.replace(/[:]/g,"%3a").replace(/[/]/g,
"%2f").replace(/[?]/g,"%3f").replace(/[/=]/g,"%3d").replace(/[&]/g,"%26");
      var tourl="https://open.weixin.qq.com/connect/oauth2/authorize?appid=
wx0c90608d772d2164&redirect_uri="+changeurl+"&response_type=code&scope=snsap
i_base&state=location#wechat_redirect";
      location.href=tourl;
    }
  }

```

```

        $("#alreadyClick").val("true");
    }

}

//重新定位位置
function locationAgain(){
    wx.getLocation({
        success: function (res) {
            var latitude = res.latitude; //纬度, 浮点数, 范围为 90 ~ -90
            var longitude = res.longitude; //经度, 浮点数, 范围为 180 ~ -180。
            var speed = res.speed; //速度, 以米/每秒计
            var accuracy = res.accuracy; //位置精度
            //alert(JSON.stringify(res));
            //转换百度坐标为腾讯坐标
            qq.maps.convertor.translate(new qq.maps.LatLng(latitude,
longitude), 1, function(res2){
                latlng = res2[0];
                var curLat=latlng.getLat();//当前的腾讯纬度
                var curLng=latlng.getLng();//当前的腾讯经度
                //保存坐标
                $("#positionX").val(curLat);
                $("#positionY").val(curLng);
                //在地图上显示
                var marker = new qq.maps.Marker({
                    map : map,
                    position : latlng
                });
                //设置当前位置
                var anchor = new qq.maps.Point(0, 0);
                var size = new qq.maps.Size(40, 45);
                var origin = new qq.maps.Point(11, 0);
                var markerIcon = new qq.maps.MarkerImage(
                    "<%=request.getContextPath()%>/image/location1.png",
                    size,null,null,size
                );
                marker.setIcon(markerIcon);
                //清空数组
                for (var i=0; i < markers.length; i++) {
                    markers[i].setMap(null);
                };
                //添加数据
                markers.push(marker);
                //地图坐标成功加载之后, 设置样式
                $("#mybutton").css("background-color", "#ffffff");
                $("#mybutton").html("打卡");
            });
        }
    });
}

```



```

    });
  }
});
}

//计算距离
var EARTH_RADIUS = 6378137.0;    //单位 M
var PI = Math.PI;

function getRad(d){
  return d*PI/180.0;
}

function getFlattnDistance(lat2,lng2){
  var lat1=parseFloat($("#hyLatitude").val());
  var lng1=parseFloat($("#hyLongitude").val());
  var f = getRad((lat1 + lat2)/2);
  var g = getRad((lat1 - lat2)/2);
  var l = getRad((lng1 - lng2)/2);

  var sg = Math.sin(g);
  var sl = Math.sin(l);
  var sf = Math.sin(f);

  var s,c,w,r,d,h1,h2;
  var a = EARTH_RADIUS;
  var fl = 1/298.257;

  sg = sg*sg;
  sl = sl*sl;
  sf = sf*sf;

  s = sg*(1-sl) + (1-sf)*sl;
  c = (1-sg)*(1-sl) + sf*sl;
  w = Math.atan(Math.sqrt(s/c));
  r = Math.sqrt(s*c)/w;
  d = 2*w*a;
  h1 = (3*r - 1)/2/c;
  h2 = (3*r + 1)/2/s;
  //alert(d+"---"+fl+"----"+h1+"-----"+sf+"-----"+sg+"-----"+h2);
  return d*(1 + fl*(h1*sf*(1-sg) - h2*(1-sf)*sg));
}

```

</script>


```

<style>
ul {
    list-style: none;
    margin: 0px;
}
</style>
</head>
<body style="overflow: hidden;" onload="init()">
<form method="post" action="registerMobileAction!register">
    <div style="width:100%;height:90%;z-index: 1" id="container"></div>
    <div style="width: 100%;height:70px;">
        <div style="width:49%;border-top:1px solid #cccccc;height:100%;
font-size: 15;color:#666666;margin-top: 2px;padding-top:16px;background-color:
#ffffff;float: left;border-right:1px solid #cccccc;" onclick="locationAgain()"
align="center">重新定位<span id="test"></span></div>
        <div id="mybutton" style="width:50%;height:100%;border-top:1px
solid #cccccc;color:#666666;font-size: 15;margin-top: 2px;padding-top:16px;
background-color:#ffffff;float: left;" onclick="registerClick()" align=
"center" >打卡</div>
    </div>
    <input type="hidden" name="hyCircle" id="hyCircle" value="{hyCircle}">
</input>
    <input type="hidden" name="hyLongitude" id="hyLongitude" value=
"{hyLongitude}"> </input>
    <input type="hidden" name="hyLatitude" id="hyLatitude" value=
"{hyLatitude}"></input>
    <input type="hidden" name="addressUrl" id="addressUrl" value=
"{addressUrl}"></input>
    <input type="hidden" name="checkInfo" id="checkInfo" value=
"{checkInfo}"></input>
    <input type="hidden" name="property.positionX" id="positionX" ></input>
    <input type="hidden" name="property.positionY" id="positionY" ></input>
    <input type="hidden" id="alreadyClick" value="false" ></input>
</form>
</body>
</html>

```

11.3.7 考勤查询 JSP 页

员工通过点击微信菜单“考勤查询”，能够查询当前月份的考勤信息，示例代码如下：

```

<%@ page language="java" contentType="text/html; charset=GBK"
    pageEncoding="GBK"%>
<!-- @author 牟云飞 -->
<html>
<head>
<script type="text/javascript" src="<%=request.getContextPath()%>/js/

```

```

jquery-1.7.js"></script>
<script type="text/javascript" src="http://res.wx.qq.com/open/js/jweixin-
1.0.0.js"></script>
<script>
    wx.config({
        debug: false, //开启调试模式,调用的所有的 API 返回值会在客户端弹出消息提示,
//输出接口信息,若要查看传入的参数,可以在 PC 端打开,参数信息会通过 log 打印出来,仅在 PC
//端时才能打印
        appId: 'wx0c90608d772d2164', //必填,企业号的唯一标识,此处填写企业号 corpid
        timestamp: "${time}", //必填,生成签名的时间戳
        nonceStr: '${nonceStr}', //必填,生成签名的随机串
        signature: '&{str1}', //必填,签名,见附录 A
        jsApiList: ['hideOptionsMenu'] //必填,需要使用的 JS 接口列表,所有 JS 接口
//列表见附录 B
    });
    wx.ready(function() {
        //config 信息验证后会执行 ready 方法,所有接口调用都必须在 config 接口获得结
//果之后。config 是一个客户端的异步操作,如果需要在页面加载时就调用相关接口,则必须把相关
//接口放在 ready 函数中调用来,以确保正确执行。对于用户触发时才调用的接口,则可以直接调用,
//不需要放在 ready 函数中。
        wx.hideOptionsMenu();
    });
</script>
<title>考勤记录</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<meta content="email=yes" name="format-detection" />
<script type="text/javascript">
    $(document).ready(function() {
        //构造当前月份的日历表格
        var regInfoStr="${regInfoStr}";
        //console.log("regInfoStr",regInfoStr);
        var thisDate = new Date();
        var vYear = thisDate.getFullYear();
        var vMon = thisDate.getMonth() + 1;
        var vDay=thisDate.getDay();
        var todayDate=thisDate.getDate();

        //获取当前月份的天数
        var allDays=0;
        var month1="4,6,9,11";
        var month2="5,7,8,10,12";
        if(vMon==2){
            if(vYear%400==0|| (vYear%4==0 && vYear%100!=0)){
                allDays=29;
            }else{
                allDays=28;
            }
        }
    });

```

```

    }
    }else if(month1.indexOf(vMon)>0){
        allDays=30;
    }else{
        allDays=31;
    }

    var today = new Array('星期日','星期一','星期二','星期三','星期四','星期五','星期六');

    var date = vMon+"/01/"+vYear;    //此处也可以写成 17/07/2014, 一样能够
    //识别。也可以写成 07-17-2014, 但需要正则转换

    var day = new Date(Date.parse(date));
    var week = today[day.getDay()];
    $("#curentDate").html(vYear+"年"+vMon+"月");
    var trObj=$("#<tr>");
    trObj.attr("height",40);
    var weekobj={"星期日":"0","星期一":"1","星期二":"2","星期三":"3","星期四":"4","星期五":"5","星期六":"6"};
    var beginWeekNum=parseInt(weekobj[week],"10");
    //如果后天的打卡信息不为空, 则显示打卡异常
    var regInfoArray="";
    if(regInfoStr!=""){
        regInfoArray=regInfoStr.split(";");
    }
    for(var i=0;i<(allDays+beginWeekNum);i++){
        //var colorStr = colorInfo.split(";")[i-beginWeekNum];
        var tdObj=$("#<td >");
        var todayColor="#666666";
        tdObj.css("color","#666666");
        tdObj.css("border-bottom","1px solid #dddddd");

        tdObj.attr("align","center");
        if(i<beginWeekNum){
            tdObj.html("&nbsp;");
        }else{
            var thisNum=i-beginWeekNum+1;
            tdObj.attr("id",thisNum+"day");
            tdObj.html(thisNum);
            //点击列时, 显示打卡详情
            tdObj.attr("onclick","showReDetail('"+thisNum+"')");
            if(regInfoArray[i-beginWeekNum]!=null){
                if(regInfoArray[i-beginWeekNum].split(":")[1]=="2"){
                    tdObj.css("background-image","url('image/ycbg.png')");
                    tdObj.css("color","#ffffff");
                }else if(regInfoArray[i-beginWeekNum].split(":")[1]=="3"){
                    tdObj.css("color","#cccccc");
                }
            }
        }
    }

```



```

        tdObj.attr("isRest","true");
        todayColor="#cccccc";
    }else if(regInfoArray[i-beginWeekNum].split(":")[1]==
"4"){
        tdObj.css("background-image","url('image/qjbg2.png')");
        tdObj.css("color","#ffffff");
    }
}
if((i-beginWeekNum+1)==todayDate){
    tdObj.css("background-image","url('image/today3.png')");
    tdObj.css("color",todayColor);
}
}

if(i>0 && (i+1)%7==0){
    trObj.append(tdObj);
    $("#weekTable").append(trObj);
    trObj=$("#<tr>");
    trObj.attr("height",40);
}else{
    trObj.append(tdObj);
}
}
//如果最后一行不够一个周期，则需要填充空的列
if((allDays+beginWeekNum)%7!=0){
    var addTdNum=7-((allDays+beginWeekNum)%7);
    for(var k=0;k<addTdNum;k++){
        var nulltdObj=$("#<td>");
        nulltdObj.css("border-bottom","1px solid #dddddd");
        nulltdObj.html("&nbsp;");
        nulltdObj.attr("align","center");
        trObj.append(nulltdObj);
    }
}
$("#weekTable").append(trObj);

//显示颜色示意
var colorTrObj=$("#<tr>");
colorTrObj.attr("height",30);
//colorTrObj.attr("bgcolor","#eaeaea");
var tdObj=$("#<td >");
tdObj.attr("colspan","7");
tdObj.css("font-size","15px");
tdObj.css("border-bottom","1px solid #dddddd");

var tdHtml="&nbsp;&nbsp;<span style=\"background-color:#F05D5D;

```



```
height:20px;width:20px;\ ">&nbspps;&nbspps;</span>&nbspps;异常&nbspps;&nbspps;
&nbspps;&nbspps;<span style=\ "background-color:#E09A5F;height:20px;width:
20px;\ ">&nbspps;&nbspps;&nbspps;</span>&nbspps;请假";
    tdObj.html(tdHtml);
    colorTrObj.append(tdObj);
    $("#weekTable").append(colorTrObj);
    //显示当前选中日期的打卡情况
    var tsTrObj=$("<tr>");
    tsTrObj.attr("height",35);
    tsTrObj.attr("id","regisRecordsTsTr");
    tsTrObj.attr("bgcolor","#f0f0f0");
    var tdHtmlTs="<td colspan='7\' style=\ "font-size:15px;border-
bottom:1px solid #dddddd;padding-left:8px;\ ">该日打卡信息为:</td>";
    tsTrObj.html(tdHtmlTs);
    $("#weekTable").append(tsTrObj);
    //显示当前选中日期的打卡情况
    var detailTrObj=$("<tr>");
    detailTrObj.attr("id","regisRecordsTr");
    var tdObjla=$("<td>");
    tdObjla.attr("id","regisRecordsTd");
    tdObjla.attr("colspan",7);
    tdObjla.css("font-size","15px");
    detailTrObj.append(tdObjla);
    $("#weekTable").append(detailTrObj);
    ajaxshowReDetail(todayDate);
});
//点击列，显示当日期的打卡详情
function showReDetail(tdobj){
    var thisDate = new Date();
    var todayDate=thisDate.getDate();
    if($("#lastSelectedTd").val()!=""&&$("#"+$("#lastSelectedTd").
val()+"day").length>0){
        $("#"+$("#lastSelectedTd").val()+"day").css("background-image",
"").css("color","#666666");
        if($("#"+$("#lastSelectedTd").val()+"day").attr("isRest")=="true"){
            $("#"+$("#lastSelectedTd").val()+"day").css("background-
image","").css("color","#cccccc");
        }
    }
    if(todayDate!=tdobj &&$("#"+tdobj+"day").css("background-image").
indexOf("qjb主g2.png")<0 &&$("#"+tdobj+"day").css("background-image").indexOf
("ycbg.png")<0){
        $("#"+tdobj+"day").css("background-image","url('image/
currentday.png')").css("color","#ffffff");
        $("#lastSelectedTd").val(tdobj);
    }
}
```

```

    ajaxshowReDetail(tdoobj);
}
//后台进行Ajax 请求数据
function ajaxshowReDetail(tdoobj){
    var thisDate = new Date()
    var vYear = thisDate.getFullYear()
    var vMon = thisDate.getMonth() + 1;
    if(vMon<10){
        vMon="0"+vMon;
    }
    if(parseInt(tdoobj,10)<10){
        tdoobj="0"+tdobj;
    }
    var queryDate=vYear+"-"+vMon+"-"+tdobj;
    var rtxCode="${userId}";
    var toUrl="<%=request.getContextPath()%>/registerMobileAction.do?action=getRecordByDate&queryDate="+queryDate+"&userId="+rtxCod;
    $.ajax({
        type: "GET",
        url: toUrl,
        data: {},
        success: function(data){
            $("#regisRecordsTd").html("");
            if(data!=""){
                $("#regisRecordsTd").attr("align","left");
                var tdHtmlUl=$("#<ul>");
                var ulHtml="";
                var dataArray=data.split(";");
                for(var i=0;i<dataArray.length;i++){
                    var regTime=dataArray[i];
                    if(regTime!=""){
                        ulHtml+="- 

```

```

    }

</script>
<style type="text/css">
ul {
    list-style: none;
    margin: 0px;
}
td{
    font-family: 宋体;
    font-size: 19px;
    font-weight: normal;
    cursor: pointer;
    background-position: center center;
    background-repeat: no-repeat;
}

</style>
</head>
<body >

    <div data-role='page' id="content_page" >
        <table id="weekTable" width="100%" style="table-layout: fixed;
border-collapse: collapse;" cellpadding="0" cellspacing="0">

            <tr height="40"><td id="curentDate" colspan="7"
align="center" style="color: #FF9E2A; border-bottom: 1px solid #dddddd; font-size:
17px;"></td>

                <tr height="40">
                    <td align="center" style="border-bottom: 1px solid
#dddddd; font-size: 15px; font-family: 微软雅黑; color: #4890F9">
                        周日
                    </td>
                    <td align="center" style="border-bottom: 1px solid
#dddddd; font-size: 15px; font-family: 微软雅黑; color: #4890F9">
                        周一
                    </td>
                    <td align="center" style="border-bottom: 1px solid
#dddddd; font-size: 15px; font-family: 微软雅黑; color: #4890F9">
                        周二
                    </td>
                    <td align="center" style="border-bottom: 1px solid
#dddddd; font-size: 15px; font-family: 微软雅黑; color: #4890F9">
                        周三
                    </td>

```



```

        <td align="center" style="border-bottom:1px solid
#dddddd;font-size:15px; font-family:微软雅黑;color:#4890F9">
                                周四
        </td>
        <td align="center" style="border-bottom:1px solid
#dddddd;font-size:15px; font-family:微软雅黑;color:#4890F9">
                                周五
        </td>
        <td align="center" style="border-bottom:1px solid
#dddddd;font-size:15px; font-family:微软雅黑;color:#4890F9">
                                周六
        </td>
    </tr>

</table>

</div>
<input type="hidden" id="lastSelectedTd" value=""/>
<input type="hidden" id="userId" value="{userId}" />
</body>
</html>

```

11.3.8 其他考勤页

安全提示页 `safePage.jsp`，主要用于非微信打开等非法访问时进入的页面，示例代码如下：

```

<%@ page language="java" contentType="text/html; charset=GBK"
    pageEncoding="GBK"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<script
    src="<%=request.getContextPath()%>/js/jquery.mobile-1.4.5.min.js">
</script>
<link
    href="<%=request.getContextPath()%>/js/jquery.mobile-1.4.5.min.css"
    rel="stylesheet" type="text/css">
<meta http-equiv="Content-Type" content="text/html; charset=GBK">
<script type="text/javascript" src="http://res.wx.qq.com/open/js/jweixin-
1.0.0.js"></script>
<meta name="viewport" content="width=device-width, initial-scale=1">
<script type="text/javascript">
    //重新打卡
    function aginpushcard(){
        location.href="<%=request.getContextPath()%>/
registerMobileAction.do?action=initPage";

```



```

    }
    wx.config({
      debug: false, //开启调试模式,调用的所有的API 返回值会在客户端弹出消息提示,
      //输出接口信息,若要查看传入的参数,可以在PC端打开,参数信息会通过log打印出来,仅在PC
      //端时才能打印
      appId: 'wx0c90608d772d2164', //必填,企业号的唯一标识,此处填写企业号 CorpID
      timestamp: "${time}", //必填,生成签名的时间戳
      nonceStr: '${nonceStr}', //必填,生成签名的随机串
      signature: '${str1}', //必填,签名,见附录A
      jsApiList: ['hideOptionsMenu'] //必填,需要使用的JS接口列表,所有JS接口
      //列表见附录B
    });
    wx.ready(function(){
      //config信息验证后会执行ready方法,所有接口调用都必须在config接口获得结
      //果之后。config是一个客户端的异步操作,如果需要在页面加载时就调用相关接口,则必须把相关
      //接口放在ready函数中调用来,以确保正确执行。对于用户触发时才调用的接口,则可以直接调用,
      //不需要放在ready函数中。
      wx.hideOptionsMenu();
    });
</script>
<title>访问失败</title>
</head>
<body>
  <div data-role="page" id="home">
    <div data-role="header">
      <meta name="viewport" content="width=device-width, initial-scale=1">
    </div>
    <div data-role="content">
      <div style="width: 100%; height: 130px; padding-top: 30px;"
        align="center">
        
      </div>
      <div
        style="width: 100%; height: 60px; font-size: 15px; font-family:
        微软雅黑; color: #444444;"
        align="center">为了保证信息安全,请勿恶意刷新数据</div>
    </div>
  </div>
</body>
</html>

```

使用谷歌等非微信内置浏览器访问时,将进入 safePage 页面,如图 11.8 所示。

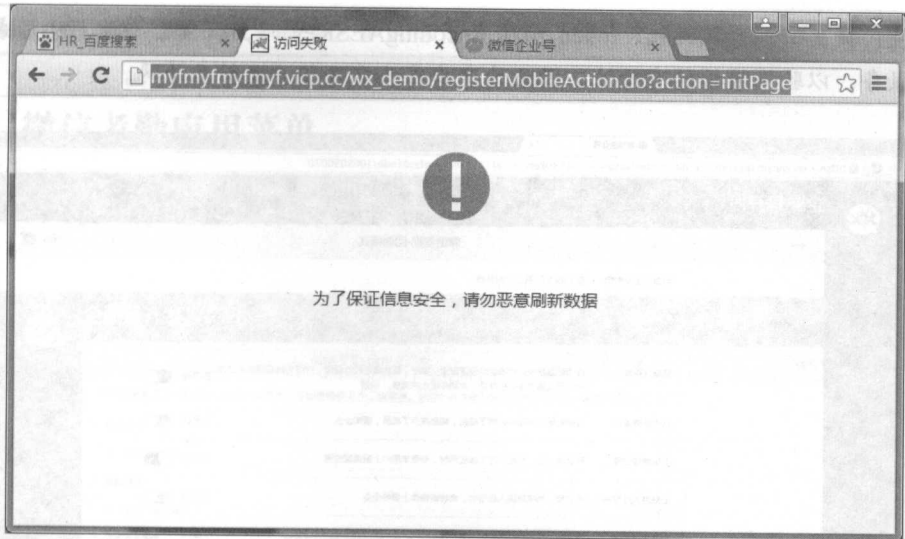


图 11.8 Google 浏览器访问

如果使用微信内置浏览器访问，则能够正常进入页面，如图 11.9 所示。

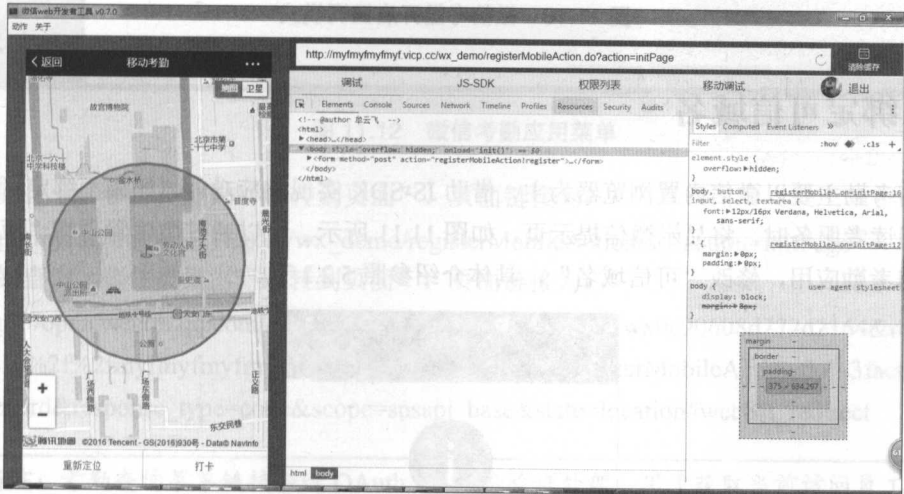


图 11.9 微信内置浏览器访问

备注：图 11.9 使用的是微信提供的微信开发工具展示，便于读者进行对比。

11.4 开启微信考勤回调模式

启动创建的 Java 工程，并将服务映射至外网（外网发布参照 2.7 节），利用 Web 服务/registerMobileServlet.slt 服务（http://域名/wx_demo/registerMobileServlet.slt）以及 WxUtil 中的 RESP_MESSAGE_TOKEN 和 RESP_MESSAGE_ENCODINGAESKEY 开启回调模式（参照 4.2 节），如图 11.10 所示。



备注：提交回调模式链接、token 以及 EncodingAESKey 时，微信需要以 Get 形式访问读者服务，以验证链接的有效性，因此读者需要开启服务并映射至外网。



图 11.10 微信考勤开启回调模式

11.5 绑定可信域名

微信考勤主要以微信内置浏览器为主，借助 JS-SDK 实现响应功能。如果不绑定可信域名，那么访问读者服务时，将显示微信提示页，如图 11.11 所示。可以通过微信管理端“应用中心”选择微信考勤应用，修改“可信域名”，具体介绍参照 5.2.1 节。



图 11.11 微信考勤可信域名



备注：域名需要进行 ICP 备案，如果仅用于开发，使用免费域名也可以。

11.6 微信考勤应用菜单

开启回调模式后，通过自定义菜单创建考勤打卡、考勤查询和使用帮助三个菜单，如图 11.12 所示。

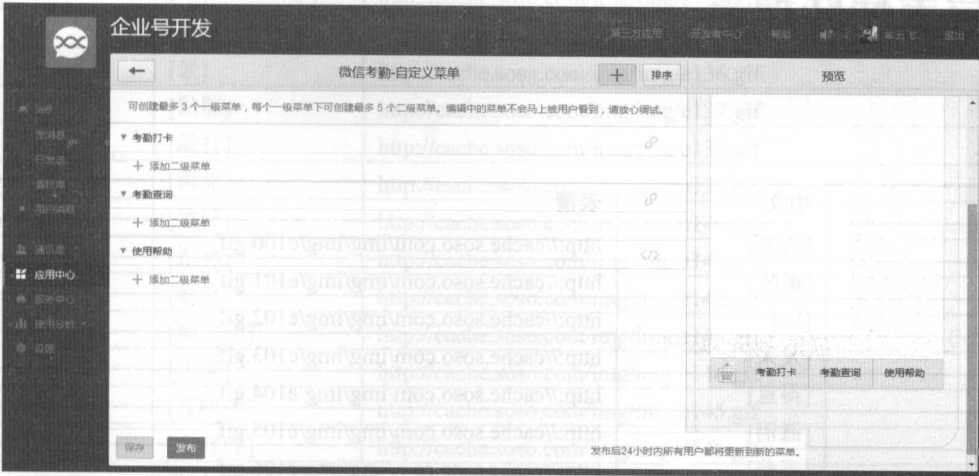


图 11.12 微信考勤应用菜单

考勤打卡，事件类型为“跳转到页面”，页面链接为：

http://myfmyfmyfmyf.vicp.cc/wx_demo/registerMobileAction.do?action=initPage

考勤查询，事件类型为“跳转到页面”，页面链接为：

https://open.weixin.qq.com/connect/oauth2/authorize?appid=wx0c90608d772d2164&redirect_uri=http%3a%2f%2fmyfmyfmyfmyf.vicp.cc%2fwx_demo%2fregisterMobileAction.do%3faction%3dretrieveRecord&response_type=code&scope=snsapi_base&state=location#wechat_redirect

备注：考勤查询菜单链接经过 OAuth 2.0 身份验证处理，用于获取当前访问员工身份，详细说明可参照 8.1 节。

使用帮助，事件类型为“菜单 key”，设置菜单事件 key 值为 KEY_HELLO，用于回调模式捕获菜单事件（参照 4.8.5 节）。

11.7 本章小结

本章在 HelleWorld 应用的基础上，实现高级应用“微信考勤”，通过企业员工日常活动开发相应的微信应用，通过 JS-API 使应用更加丰富，提升视觉体验。希望读者通过本章的案例能够掌握微信企业号开发的方法，从而创建更加丰富的微信企业号应用。

附录 A

微信表情转换表

文本	中文	表情
/::)	[微笑]	http://cache.soso.com/img/img/e100.gif
/::~~	[撇嘴]	http://cache.soso.com/img/img/e101.gif
/::B	[色]	http://cache.soso.com/img/img/e102.gif
/::	[发呆]	http://cache.soso.com/img/img/e103.gif
/:8~)	[得意]	http://cache.soso.com/img/img/e104.gif
/::<	[流泪]	http://cache.soso.com/img/img/e105.gif
/::\$	[害羞]	http://cache.soso.com/img/img/e106.gif
/::X	[闭嘴]	http://cache.soso.com/img/img/e107.gif
/::Z	[睡]	http://cache.soso.com/img/img/e108.gif
/::'	[大哭]	http://cache.soso.com/img/img/e109.gif
/::	[尴尬]	http://cache.soso.com/img/img/e110.gif
/::@	[发怒]	http://cache.soso.com/img/img/e111.gif
/::P	[调皮]	http://cache.soso.com/img/img/e112.gif
/::D	[呲牙]	http://cache.soso.com/img/img/e113.gif
/::O	[惊讶]	http://cache.soso.com/img/img/e114.gif
/::('	[难过]	http://cache.soso.com/img/img/e115.gif
/::+	[酷]	http://cache.soso.com/img/img/e116.gif
/:--b	[冷汗]	http://cache.soso.com/img/img/e117.gif
/::Q	[抓狂]	http://cache.soso.com/img/img/e118.gif
/::T	[吐]	http://cache.soso.com/img/img/e119.gif
/:,@P	[偷笑]	http://cache.soso.com/img/img/e120.gif
/:,@-D	[可爱]	http://cache.soso.com/img/img/e121.gif
/::d	[白眼]	http://cache.soso.com/img/img/e122.gif
/:,@o	[傲慢]	http://cache.soso.com/img/img/e123.gif
/::g	[饥饿]	http://cache.soso.com/img/img/e124.gif
/:-)	[困]	http://cache.soso.com/img/img/e125.gif
/::!	[惊恐]	http://cache.soso.com/img/img/e126.gif
/::L	[流汗]	http://cache.soso.com/img/img/e127.gif
/::>	[憨笑]	http://cache.soso.com/img/img/e128.gif
/::,@	[大兵]	http://cache.soso.com/img/img/e129.gif

续表

文本	中文	表情
/:,@f	[奋斗]	http://cache.soso.com/img/img/e130.gif
/:-S	[咒骂]	http://cache.soso.com/img/img/e131.gif
/:?	[疑问]	http://cache.soso.com/img/img/e132.gif
/:,@x	[嘘]	http://cache.soso.com/img/img/e133.gif
/:,@@	[晕]	http://cache.soso.com/img/img/e134.gif
/::8	[折磨]	http://cache.soso.com/img/img/e135.gif
/:,@!	[衰]	http://cache.soso.com/img/img/e136.gif
/:!!!	[骷髅]	http://cache.soso.com/img/img/e137.gif
/:xx	[敲打]	http://cache.soso.com/img/img/e138.gif
/:bye	[再见]	http://cache.soso.com/img/img/e139.gif
/:wipe	[擦汗]	http://cache.soso.com/img/img/e140.gif
/:dig	[抠鼻]	http://cache.soso.com/img/img/e141.gif
/:handclap	[鼓掌]	http://cache.soso.com/img/img/e142.gif
/:&-([糗大了]	http://cache.soso.com/img/img/e143.gif
/:B-)	[坏笑]	http://cache.soso.com/img/img/e144.gif
/:<@	[左哼哼]	http://cache.soso.com/img/img/e145.gif
/:@>	[右哼哼]	http://cache.soso.com/img/img/e146.gif
/::O	[哈欠]	http://cache.soso.com/img/img/e147.gif
/:->	[鄙视]	http://cache.soso.com/img/img/e148.gif
/:P-([委屈]	http://cache.soso.com/img/img/e149.gif
/:	[快哭了]	http://cache.soso.com/img/img/e150.gif
/:X-)	[阴险]	http://cache.soso.com/img/img/e151.gif
/::*	[亲亲]	http://cache.soso.com/img/img/e152.gif
/:@x	[吓]	http://cache.soso.com/img/img/e153.gif
/:8*	[可怜]	http://cache.soso.com/img/img/e154.gif
/:pd	[菜刀]	http://cache.soso.com/img/img/e155.gif
/:<W>	[西瓜]	http://cache.soso.com/img/img/e156.gif
/:beer	[啤酒]	http://cache.soso.com/img/img/e157.gif
/:basketb	[篮球]	http://cache.soso.com/img/img/e158.gif
/:oo	[乒乓]	http://cache.soso.com/img/img/e159.gif
/:coffee	[咖啡]	http://cache.soso.com/img/img/e160.gif
/:eat	[饭]	http://cache.soso.com/img/img/e161.gif
/:pig	[猪头]	http://cache.soso.com/img/img/e162.gif
/:rose	[玫瑰]	http://cache.soso.com/img/img/e163.gif
/:fade	[凋谢]	http://cache.soso.com/img/img/e164.gif
/:showlove	[嘴唇]	http://cache.soso.com/img/img/e165.gif
/:heart	[爱心]	http://cache.soso.com/img/img/e166.gif
/:break	[心碎]	http://cache.soso.com/img/img/e167.gif
/:cake	[蛋糕]	http://cache.soso.com/img/img/e168.gif
/:li	[闪电]	http://cache.soso.com/img/img/e169.gif

续表

文本	中文	表情
/:bome	[炸弹]	http://cache.soso.com/img/img/e170.gif
/:kn	[刀]	http://cache.soso.com/img/img/e171.gif
/:footb	[足球]	http://cache.soso.com/img/img/e172.gif
/:ladybug	[瓢虫]	http://cache.soso.com/img/img/e173.gif
/:shit	[便便]	http://cache.soso.com/img/img/e174.gif
/:moon	[月亮]	http://cache.soso.com/img/img/e175.gif
/:sun	[太阳]	http://cache.soso.com/img/img/e176.gif
/:gift	[礼物]	http://cache.soso.com/img/img/e177.gif
/:hug	[拥抱]	http://cache.soso.com/img/img/e178.gif
/:strong	[强]	http://cache.soso.com/img/img/e179.gif
/:weak	[弱]	http://cache.soso.com/img/img/e180.gif
/:share	[握手]	http://cache.soso.com/img/img/e181.gif
/:v	[胜利]	http://cache.soso.com/img/img/e182.gif
/:@)	[抱拳]	http://cache.soso.com/img/img/e183.gif
/:jj	[勾引]	http://cache.soso.com/img/img/e184.gif
/:@@	[拳头]	http://cache.soso.com/img/img/e185.gif
/:bad	[差劲]	http://cache.soso.com/img/img/e186.gif
/:lvu	[爱你]	http://cache.soso.com/img/img/e187.gif
/:no	[NO]	http://cache.soso.com/img/img/e188.gif
/:ok	[OK]	http://cache.soso.com/img/img/e189.gif
/:love	[爱情]	http://cache.soso.com/img/img/e190.gif
/:<L>	[飞吻]	http://cache.soso.com/img/img/e191.gif
/:jump	[跳跳]	http://cache.soso.com/img/img/e192.gif
/:shake	[发抖]	http://cache.soso.com/img/img/e193.gif
/:<O>	[上火]	http://cache.soso.com/img/img/e194.gif
/:circle	[转圈]	http://cache.soso.com/img/img/e195.gif
/:kotow	[磕头]	http://cache.soso.com/img/img/e196.gif
/:turn	[回头]	http://cache.soso.com/img/img/e197.gif
/:skip	[跳绳]	http://cache.soso.com/img/img/e198.gif
/:oY	[挥手]	http://cache.soso.com/img/img/e199.gif
/:#-0	[激动]	http://cache.soso.com/img/img/e200.gif
/:hiphot	[街舞]	http://cache.soso.com/img/img/e201.gif
/:kiss	[献吻]	http://cache.soso.com/img/img/e202.gif
/:<&	[左太极]	http://cache.soso.com/img/img/e203.gif
/:&>	[右太极]	http://cache.soso.com/img/img/e204.gif

返回码说明表

返回码	说明
-1	系统繁忙
0	请求成功
40001	获取access_token时Secret错误，或者access_token无效
40002	不合法的凭证类型
40003	不合法的UserID
40004	不合法的媒体文件类型
40005	不合法的文件类型
40006	不合法的文件大小
40007	不合法的媒体文件ID
40008	不合法的消息类型
40013	不合法的CorpID
40014	不合法的access_token
40015	不合法的菜单类型
40016	不合法的按钮个数
40017	不合法的按钮类型
40018	不合法的按钮名字长度
40019	不合法的按钮Key长度
40020	不合法的按钮URL长度
40021	不合法的菜单版本号
40022	不合法的子菜单级数
40023	不合法的子菜单按钮个数
40024	不合法的子菜单按钮类型
40025	不合法的子菜单按钮名字长度
40026	不合法的子菜单按钮Key长度
40027	不合法的子菜单按钮URL长度
40028	不合法的自定义菜单使用成员
40029	不合法的oauth_code
40031	不合法的UserID列表
40032	不合法的UserID列表长度
40033	不合法的请求字符，不能包含\uxxxx格式的字符
40035	不合法的参数

续表

返回码	说明
40038	不合法的请求格式
40039	不合法的URL长度
40040	不合法的插件token
40041	不合法的插件ID
40042	不合法的插件会话
40048	URL中包含不合法domain
40054	不合法的子菜单URL域名
40055	不合法的按钮URL域名
40056	不合法的AgentID
40057	不合法的callbackurl或者callbackurl验证失败
40058	不合法的红包参数
40059	不合法的上报地理位置标志位
40060	设置上报地理位置标志位时没有设置callbackurl
40061	设置应用头像失败
40062	不合法的应用模式
40063	参数为空
40064	管理组名字已存在
40065	不合法的管理组名字长度
40066	不合法的部门列表
40067	标题长度不合法
40068	不合法的标签ID
40069	不合法的标签ID列表
40070	列表中所有标签（成员）ID都不合法
40071	不合法的标签名字，标签名字已经存在
40072	不合法的标签名字长度
40073	不合法的OpenID
40074	news消息不支持指定为高保密消息
40077	不合法的预授权码
40078	不合法的临时授权码
40079	不合法的授权信息
40080	不合法的suitesecret
40082	不合法的suitetoken
40083	不合法的suiteID
40084	不合法的永久授权码
40085	不合法的suiteticket
40086	不合法的第三方应用AppID
40092	导入文件存在不合法的内容
40093	不合法的跳转target
40094	不合法的URL
40095	修改失败，并发冲突
41001	缺少access_token参数
41002	缺少CorpID参数

续表

返回码	说明
41003	缺少refresh_token参数
41004	缺少secret参数
41005	缺少多媒体文件数据
41006	缺少media_id参数
41007	缺少子菜单数据
41008	缺少oauth_code
41009	缺少UserID
41010	缺少URL
41011	缺少AgentID
41012	缺少应用头像MediaID
41013	缺少应用名字
41014	缺少应用描述
41015	缺少Content
41016	缺少标题
41017	缺少标签ID
41018	缺少标签名字
41021	缺少SuiteID
41022	缺少suitetoken
41023	缺少suiteticket
41024	缺少suitesecret
41025	缺少永久授权码
41034	缺少login_ticket
41035	缺少跳转target
42001	access_token过期
42002	refresh_token过期
42003	oauth_code过期
42004	插件token过期
42007	预授权码失效
42008	临时授权码失效
42009	suitetoken失效
43001	需要GET请求
43002	需要POST请求
43003	需要HTTPS
43004	需要成员已关注
43005	需要好友关系
43006	需要订阅
43007	需要授权
43008	需要支付授权
43010	需要处于回调模式
43011	需要企业授权
43013	应用对成员不可见
44001	多媒体文件为空

续表

返回码	说明
44002	POST的数据包为空
44003	图文消息内容为空
44004	文本消息内容为空
45001	多媒体文件大小超过限制
45002	消息内容大小超过限制
45003	标题大小超过限制
45004	描述大小超过限制
45005	链接长度超过限制
45006	图片链接长度超过限制
45007	语音播放时间超过限制
45008	图文消息的文章数量不能超过10条
45009	接口调用超过限制
45010	创建菜单个数超过限制
45015	回复时间超过限制
45016	系统分组，不允许修改
45017	分组名字过长
45018	分组数量超过上限
45022	应用名字长度不合法，合法长度为2~16个字
45024	账号数量超过上限
45025	同一个成员每周只能邀请一次
45026	触发删除用户数的保护
45027	mpnews每天只能发送100次
45028	素材数量超过上限
45029	media_id对该应用不可见
45032	作者名字长度超过限制
46001	不存在媒体数据
46002	不存在的菜单版本
46003	不存在的菜单数据
46004	不存在的成员
47001	解析JSON/XML内容错误
48001	API未授权
48002	API禁用(一般是管理组类型与API不匹配，例如普通管理组调用会话服务的API)
48003	suitetoken无效
48004	授权关系无效
48005	API已废弃
50001	redirect_uri未授权
50002	成员不在权限范围
50003	应用已停用
50004	成员状态不正确，需要成员为企业验证中状态
50005	企业已禁用
60001	部门长度不符合限制
60002	部门层级深度超过限制

续表

返回码	说明
60003	部门不存在
60004	父部门不存在
60005	不允许删除有成员的部门
60006	不允许删除有子部门的部门
60007	不允许删除根部门
60008	部门ID或者部门名称已存在
60009	部门名称含有非法字符
60010	部门存在循环关系
60011	管理组权限不足，（user/department/agent）无权限
60012	不允许删除默认应用
60013	不允许关闭应用
60014	不允许开启应用
60015	不允许修改默认应用可见范围
60016	不允许删除存在成员的标签
60017	不允许设置企业
60019	不允许设置应用地理位置上报开关
60020	访问ip不在白名单之中
60025	主页型应用不支持的消息类型
60027	不支持第三方修改主页型应用字段
60028	应用已授权予第三方，不允许通过接口修改主页URL
60029	应用已授权予第三方，不允许通过接口修改可信域名
60102	UserID已存在
60103	手机号码不合法
60104	手机号码已存在
60105	邮箱不合法
60106	邮箱已存在
60107	微信号不合法
60108	微信号已存在
60109	QQ号已存在
60110	用户同时归属部门超过20个
60111	UserID不存在
60112	成员姓名不合法
60113	身份认证信息（微信号/手机/邮箱）不能同时为空
60114	性别不合法
60115	已关注成员微信不能修改
60116	扩展属性已存在
60118	成员无有效邀请字段，详情参考(邀请成员关注)的接口说明
60119	成员已关注
60120	成员已禁用
60121	找不到该成员
60122	邮箱已被外部管理员使用
60123	无效的部门ID

返回码	说明
60124	无效的父部门ID
60125	非法部门名字, 长度超过限制、重名等
60126	创建部门失败
60127	缺少部门ID
60128	字段不合法, 可能存在主键冲突或者格式错误
60129	用户设置了拒绝邀请
80001	可信域名不匹配, 或者可信域名没有IPC备案 (后续将不能在该域名下正常使用js-sdk)
81003	邀请额度已用完
81004	部门数量超过上限
82001	发送消息或者邀请的参数全部为空或者全部不合法
82002	不合法的PartyID列表长度
82003	不合法的TagID列表长度
82004	微信版本号过低
85002	包含不合法的词语
86001	不合法的会话ID
86003	不存在的会话ID
86004	不合法的会话名
86005	不合法的会话管理员
86006	不合法的成员列表大小
86007	不存在的成员
86101	需要会话管理员权限
86201	缺少会话ID
86202	缺少会话名
86203	缺少会话管理员
86204	缺少成员
86205	非法的会话ID长度
86206	非法的会话ID数值
86207	会话管理员不在用户列表中
86208	消息服务未开启
86209	缺少操作者
86210	缺少会话参数
86211	缺少会话类型 (单聊或者多聊)
86213	缺少发件人
86214	非法的会话类型
86215	会话已存在
86216	非法会话成员
86217	会话操作者不在成员列表中
86218	非法会话发件人
86219	非法会话收件人
86220	非法会话操作者
86221	单聊模式下, 发件人与收件人不能为同一人
86222	不允许消息服务访问的API

续表

返回码	说明
86304	不合法的消息类型
86305	客服服务未启用
86306	缺少发送人
86307	缺少发送人类型
86308	缺少发送人ID
86309	缺少接收人
86310	缺少接收人类型
86311	缺少接收人ID
86312	缺少消息类型
86313	缺少客服, 发送人或接收人类型, 必须有一个为kf
86314	客服不唯一, 发送人或接收人类型, 必须只有一个为kf
86315	不合法的发送人类型
86316	不合法的发送人ID。UserID不存在、OpenID不存在、kf不存在
86317	不合法的接收人类型
86318	不合法的接收人ID。UserID不存在、OpenID不存在、kf不存在
86319	不合法的客服, kf不在客服列表中
86320	不合法的客服类型
88001	缺少seq参数
88002	缺少offset参数
88003	非法seq
90001	未认证摇一摇周边
90002	缺少摇一摇周边ticket参数
90003	摇一摇周边ticket参数不合法
90004	摇一摇周边ticket过期
90005	未开启摇一摇周边服务
90005	未开启摇一摇周边服务
91004	卡券已被核销
91011	无效的code
91014	缺少卡券详情
91015	代金券缺少least_cost或者reduce_cost参数
91016	折扣券缺少discount参数
91017	礼品券缺少gift参数
91019	缺少卡券sku参数
91020	缺少卡券有效期
91021	缺少卡券有效期类型
91022	缺少卡券logo_url
91023	缺少卡券code类型
91025	缺少卡券title
91026	缺少卡券color
91027	缺少offset参数
91028	缺少count参数
91029	缺少card_id

续表

返回码	说明
91030	缺少卡券code
91031	缺少卡券notice
91032	缺少卡券description
91033	缺少ticket类型
91036	不合法的有效期
91038	变更库存值不合法
91039	不合法的卡券ID
91040	不合法的ticket type
91041	没有创建, 上传卡券logo, 以及核销卡券的权限
91042	没有该卡券投放权限
91043	没有修改或者删除该卡券的权限
91044	不合法的卡券参数
91045	缺少团购券groupon结构
91046	缺少现金券cash结构
91047	缺少折扣券discount 结构
91048	缺少礼品券gift结构
91049	缺少优惠券coupon结构
91050	缺少卡券必填字段
91051	商户名称超过12个汉字
91052	卡券标题超过9个汉字
91051	商户名称超过12个汉字
91053	卡券提醒超过16个汉字
91054	卡券描述超过1024个汉字
91055	卡券副标题长度超过18个汉字

本书主要通过实际案例，深入浅出，一步步带领您学习微信开发企业号开发，实现各类客户需求。在微信流行的移动互联网时代，掌握一门轻应用开发技术，更加有利于提高自身价值，在IT浪潮中占据主导权。

关注作者：





微信企业号开发 完全自学手册

微信企业号是微信平台为企业用户提供的轻应用入口，它具有跨系统平台、易于部署推广的显著特点。本书作者具有丰富的移动应用和微信应用的开发经验，有这本实战指南在手，相信您一定能快速构建出令人满意的企业号应用，从而助力企业迅速实现移动化的目标。

烟台海颐软件股份有限公司 李锐

移动互联网大潮已经来袭，改变了人们的生活习惯，微信已成为社交、营销的重要工具。微信也是企业和公司的重要营销、宣传渠道。微信公众号的开发、维护、经营的优劣，决定着公司在这个渠道上的竞争力。20世纪90年代，错过了互联网爆发带来的财富机遇，现在微信的爆发期来了，您还想错过吗？

杰瑞教育科技有限公司创始人&总经理 杨延成

微信作为主流社交方式，不知不觉间，已成为我们生活中必不可少的一部分，目前微信活跃用户数已超过8亿，基于微信的开发正在被越来越多的人所关注。本书系统地介绍了微信企业号开发方法，有助于快速学习和掌握微信企业号开发，满足企业移动办公应用要求。

烟台华东数据科技有限公司 刘澎

系统，全面，由浅入深，循序渐进，将微信企业号开发过程中遇到的各种问题结合实例进行讲解，给人一种茅塞顿开的感觉，是一本不可多得的微信企业号开发教程，值得学习。

山东大学 鲁燃

从大处着眼，从小处着手。本书以开发高效微信企业号应用为目标，层层分解、环环相扣，内容覆盖微信企业号开发从注册到部署的各项技术，字里行间均体现出强烈的实战思维，真正是看得懂、学得会、用得着。

山东中医药大学 马金刚



博文视点Broadview



@博文视点Broadview



策划编辑：张月萍
责任编辑：安娜
封面设计：李玲

上架建议：计算机\程序开发

ISBN 978-7-121-30809-3



定价：76.00元